VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

MODELOVÁNÍ NA ZÁKLADĚ DAT Z ARCHIVÁLIÍ

MODELING BASED ON ARCHIVE DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN MIKAN**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. FRANTIŠEK ZBOŘIL, Ph.D.**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Mikan Martin**
Obor: Informační technologie
Téma: **Modelování na základně dat z archiválií**
Modelling on Historical Data
Kategorie: Modelování a simulace

Pokyny:

1. Seznamte se se současnými systémy které mají vztah ke zpracování historických dat z archiválií
2. Navrhněte systém, který by umožňoval na základě takovýchto opsaných dat vytvářet sociální modely. Vymezte role osob, které se pro jednotlivé typy záznamů objevují a navrhněte, jak lze uživatelsky přívětivě tyto role přisuzovat objektům vytvářeného modelu.
3. Realizujte takový systém a pro poskytnutá data (v rozsahu 1 farnosti a 200 let) vytvořte několik modelů, například rodokmenů některého z rodů.
4. Diskutujte dosažené výsledky a navrhněte další možná rozšíření takového systému.

Literatura:

- Lednická, B.: Sestavte si rodokmen, pátráme po svých předcích, Grada, 2012

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zbořil František, doc. Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Předmětem této bakalářské práce je návrh a implementace GUI aplikace, která má za úkol prezentovat dostupná data z archiválií, přiřadit těmto datům role a následně nad těmito daty vygenerovat vybraný sociální model. Nejprve byla provedene analýza již existujících aplikací a analýza dostupných technologií pro vývoj této aplikace. Výsledkem této práce je GUI aplikace určená pro desktop systémy, která umožňuje výběr osob z tabulky, přiřazení rolí těmto osobám a vygenerování rodokmenu, či rozdělení vybraných osob podle pohlaví.

Abstract

This bachelor thesis focuses on design and implementation of a GUI application, which is supposed to present available data from archival materials, assign roles to the mentioned data and generate a certain social model using those data. The first step was analysing already existing applications and available technologies for developping the application. The product of this thesis is a GUI application created for desktop systems which allows selection of a person from a table, assigning roles to these people and generating a family tree or a gender breakdown model using the people mentioned above.

Klíčová slova

genealogie, Java, JavaFX, MySQL, archiválie, GUI, socální modely, vizualizace dat

Keywords

genealogy, Java, JavaFX, MySQL, archival materials, GUI, social models, data visualization

Citace

MIKAN, Martin. *Modelování na základě dat z archiválií*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. František Zbořil, Ph.D.

Modelování na základě dat z archiválií

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Františka Zbořila, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Mikan
28. května 2018

Poděkování

Rád bych poděkoval panu Doc. Ing. Františkovi Zbořilovi, Ph.D. za vedení bakalářské práce, cenné rady a poskytnuté materiály a konzultace.

Obsah

Kapitola 1

Úvod

Hlavním cílem této práce je návrh a následná implementace aplikace v programovacím jazyce Java. Aplikace bude sloužit k zpracování dat z opsaných matrik a podobných archiválií a následném modelování (různých sociálních modelů) nad předem specifikovanými záznamy. Uživatel také bude mít možnost vyfiltrovat jím specifikované objekty, ze kterých se následně budou tvořit modely.

Tvorba této aplikace je motivována především neexistencí jiné aplikace umožňující vytváření takovýchto modelů nad velmi nestrukturovanými daty z matrik a vůbec zjištění jakým nejlepším způsobem zpracovávat tato data.

Práce je rozdělená do několika kapitol postupně popisující danou problematiku, návrh řešení a implementaci s postupem, který vedl k výslednému stavu aplikace. V kapitole 1 jsou rozebrány již existující aplikace a jejich přínos. Kapitola 2 se zaměřuje na analýzu požadavků na aplikaci a rozebrání hlavních cílů aplikace, popis cílové skupiny a dalších témat. V kapitole 3 budou rozebrány použité technologie, které nám umožní nahlédnout do procesu vytváření aplikaci a případných softwarových nedostatků jednotlivých technologií. V kapitole 4 bude rozebrán proces návrhu aplikace, bude zde popsán návrh databáze, základní struktury, které musí figurovat v aplikaci, a další. V kapitole 5 bude podrobně rozepsána samotná implementace aplikace – struktura aplikace, myšlenkové pochody a proces vytváření aplikace krok po kroku. V kapitole 6 bude popsán proces testování a úpravy, které vzešly z testování aplikace. V kapitole 7 budou rozebrány možná rozšíření aplikace. V kapitole 8 se nachází závěr spolu se stručným zhodnocením výsledků vzhledem k cílům práce.

Kapitola 2

Analýza existujících aplikací

Po důsledném hledání existujících aplikací, které mají vztah ke zpracování historických dat z archiválií a jejich následného modelování jsem spolu s archiváři z MUNI dospěl k závěru, že takové systémy v době psaní práce neexistují, anebo se velmi obtížně dohledávají.

I přesto v této kapitole popíši některé nejznámější genealogické aplikace, jaké jsou jejich funkce a rozdíly mezi mnou vytvářenou aplikací.

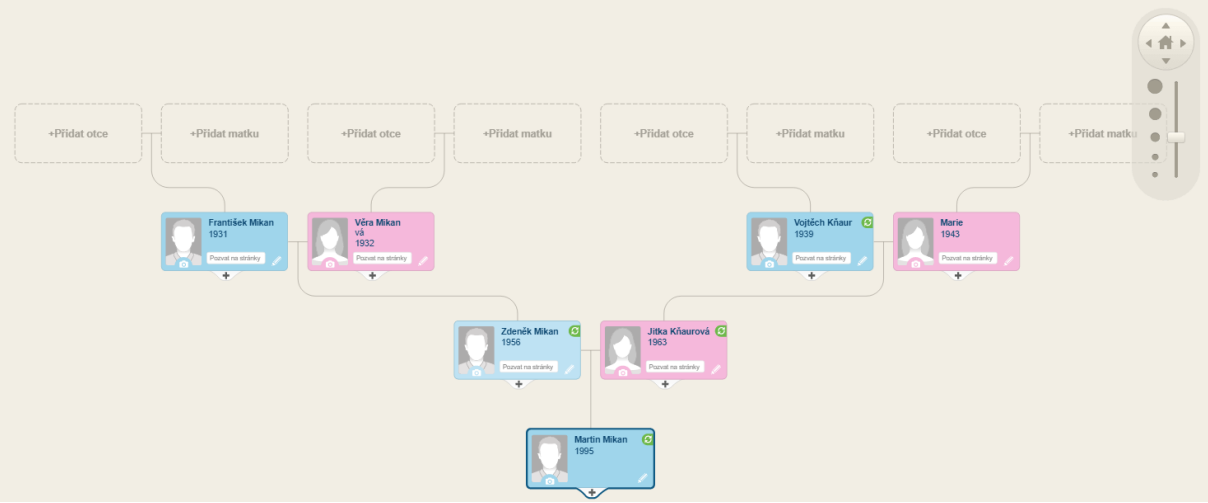
2.1 MyHeritage

První aplikací, kterou budu v této kapitole rozebírat, je MyHeritage. MyHeritage, na rozdíl od aplikací, ve kterých lze pouze vytvořit strom a pracovat nad ním, funguje spíše jako sociální síť. Umožňuje vytvořit rodokmen, který je veřejně dostupný a do kterého lze zvát jeho členy.

Pomocí tohoto rodokmenu je možné sdílet fotografie, vyhledávat předky, plánovat události. Jednotlivé osoby mohou také sdílet videa, email a dokonce životopis.

Společnost také udržuje velkou interní databázi osob spolu s možností vyhledávání předků ve velkém množství světových databází.

MyHeritage je v současné době druhou největší společností zabývající se genealogií.



Obrázek 2.1: Screenshot tvoření rodokmenu v MyHeritage, obrázek převzat z [6]

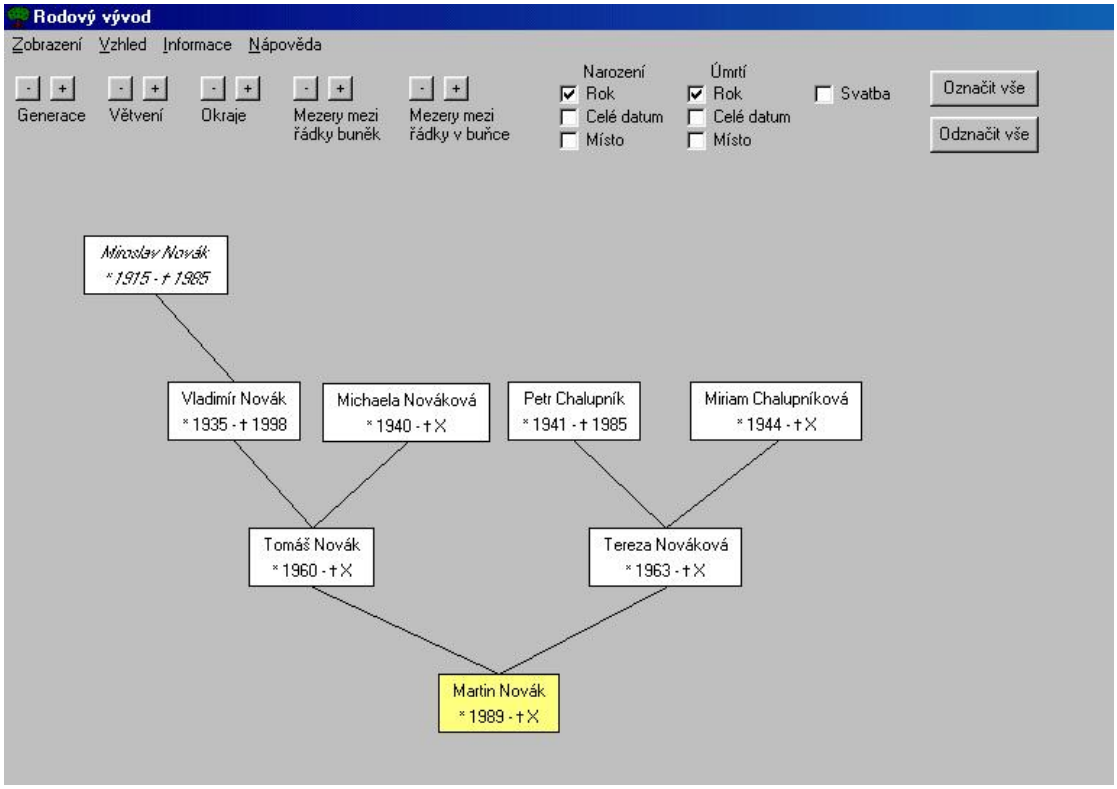
2.2 Ancestry

Druhá aplikace je česká aplikace „Ancestry“, dříve známá jako „Rodokmen“. Tato aplikace umožňuje ukládat informace o různých členech vašeho rodu v přehledné formě. Základní informace, které bere v potaz, jsou jméno, příjmení, datum narození a další, ale je možné si k jednotlivým osobám také uložit informace typu vzdělání, telefonní číslo nebo i obrázky a zvukové záznamy.

Nad tímto vytvořeným stromem poté umí program vytvářet statistiky celého rodu (nebo konkrétní osoby), vykreslit rodokmen a vygenerovat příbuzenské stromy (rodový vývod, rozrod rodu, . . .). Program také umožňuje nalezení vztahu mezi osobami (bratranec, kmotr a další).

Vygenerovaný strom lze také uložit v PDF, JPG, SVG či převést do HTML.

Hlavním rozdílem mezi touto aplikací a mnou vytvářenou aplikací je samozřejmě nutnost vložit data, se kterými pracujeme ručně.



Obrázek 2.2: Screenshot tvoření rodokmenu v Ancestry, obrázek převzat z [1]

Kapitola 3

Analýza požadavků

V této kapitole budou popsány a postupně shrnuty požadavky na výslednou aplikaci a odůvodnění těchto požadavků. Následně bude vysvětlena hlavní motivace tvorby aplikace a přístup k problematice vývoje.

3.1 Popis problému

Hlavním problémem, který motivoval tvorbu této aplikace, je neexistence takovýchto aplikací obecně. Důvod lze sledovat v tom, že v této době se teprve pomalu přepisují data z matrik do nějaké elektronické podoby, která by pak v budoucnu mohla být použita v aplikacích právě jako tato.

Systémy, které doposud vznikly, dovolují ručně vytvářet rodokmen, avšak nedokáží jej automaticky generovat z poskytnutých dat. Toto je na rozdíl od těchto aplikací cílem mnou vytvářeného systému.

3.2 Požadavky na aplikaci

Tato aplikace je určena pro širokou veřejnost, neexistuje tedy konkrétní cílová skupina uživatelů. Z toho plyne, že používání a manipulace s touto aplikací by mělo být co nejintuitivnější, přehledné a bezproblémové, aby kdokoliv, kdo bude mít zájem o práci s touto aplikací, neměl zbytečné problémy v případě neznalosti počítačových aplikací. Také s ohledem na dostupnost byl zvolen programovací jazyk Java. Klíčové prvky a požadavky lze shrnout do následujících bodů:

- Návrh a implementace databáze
- Zpracování poskytnutých dat z matrik
- Možnost filtrace připravovaných dat
- Přívětivý a praktický vzhled aplikace
- Jednoduché, intuitivní a přehledné ovládání
- Případně výběr a generování modelů
- Vizuální zobrazení vybraných dat

3.2.1 Uživatelské rozhraní

Důraz je především kladem na jednoduché a přívětivé ovládání aplikace. Měla by sloužit převážně k vybrání námi požadovaných dat, přiřazení jednotlivých rolí těmto objektům, filtrování těchto objektů a následnému vygenerování modelu. Aplikace by neměla obsahovat žádné rušivé prvky. Bude implementovat základní prvky pro interakci s uživatelem.

3.2.2 Databáze

Pro požadovanou funkci aplikace je potřeba mít námi požadovaná data uložena ve formě databáze ve strukturované formě, která nám ulehčí příslušné modelování. Je nutné zvolit vhodnou platformu a databázový server a poté buďto ručně, nebo pomocí scriptu, naplnit databázi potřebnými daty za účelem demonstrace funkčnosti aplikace.

Kapitola 4

Použité technologie

V této kapitole budou popsány použité technologie, budou zde uvedeny důvody pro jejich vybrání, v čem vyhovují našim požadavkům a jejich případné nedostatky.

4.1 Programovací jazyk

Samotná aplikace je implementována v programovacím jazyce Java. Java je programovací jazyk soustředěný na objektově orientovaný návrh, vyvinutý firmou Sun Microsystems 23. května 1995 [2].

Hlavní důvody pro použití tohoto jazyka jsou jednoduchost, výše zmíněná soustředěnost na objektově orientovaný návrh, který usnadní případné rozšíření tohoto projektu v budoucnosti, a přenositelnost aplikací na velké množství zařízení.

Samotný projekt je vytvořený na softwarové platformě JavaFX. JavaFX je framework soustředící se na tvorbu „bohatých“ aplikací. Výsledná aplikace dokáže být velmi líbivá a zároveň je kladen velký důraz na jednoduchost její tvorby [7]. Jedna z hlavních výhod je používání souborů v jazyce FXML.

FXML je jazyk založený na XML vytvořený firmou Oracle Corporation, který je používán v JavaFX na vytvoření grafického uživatelského rozhraní. Tímto je nabídnuta možnost naprosto oddělit logiku aplikace od uživatelského grafického rozhraní. Struktura uživatelského rozhraní je také mnohem přehlednější a lehce upravitelná.

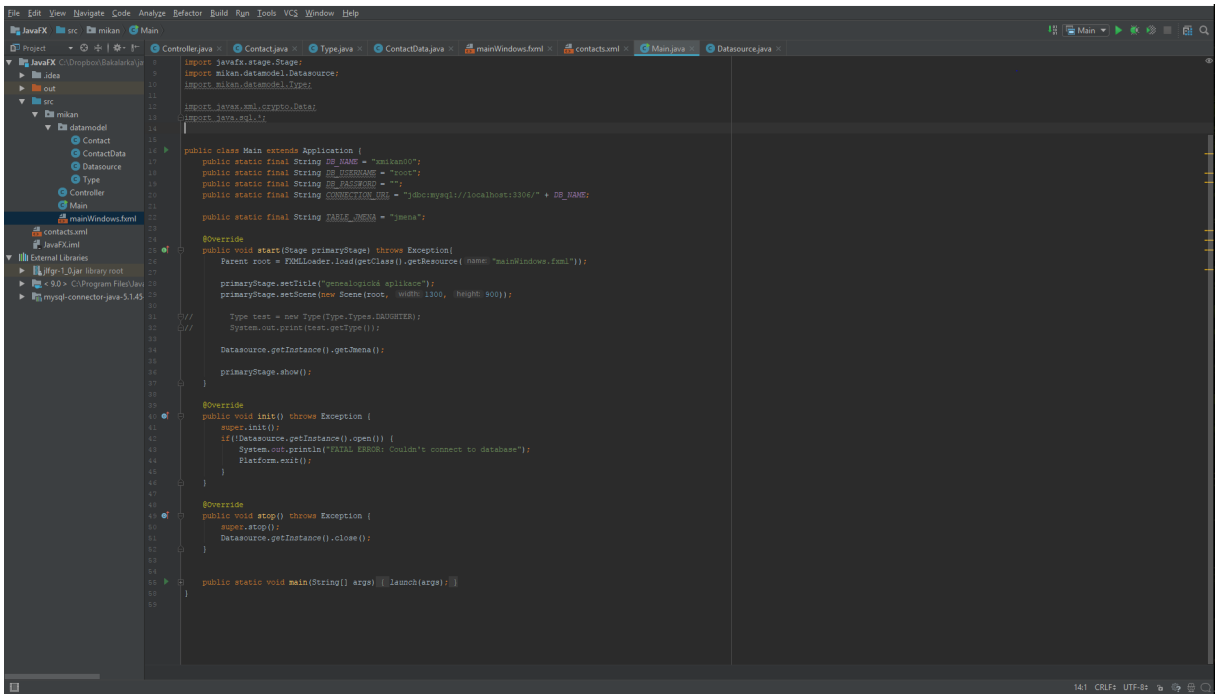
Spolu s FXML soubory jde ruku v ruce aplikace Scene Builder. Scene Builder je GUI aplikace, která spolupracuje s JavaFX a umožňuje vytvářet FXML soubory pomocí jednoduchého a přehledného drag and drop uživatelského rozhraní. Scene Builder byl, ale již není, spravován firmou Oracle Corporation. Vývoj tohoto nástroje převzala firma Gluon.

4.2 Vývojové prostředí

Při vývoji aplikace bylo použito vývojové prostředí IntelliJ IDEA. Je to komerční vývojové prostředí od firmy JetBrains soustředěno na vývoj v Javě. Nabízí nadprůměrnou detekci duplikovaného kódu, propracované doplňování kódu, detekci chyb, debuggovací prostředí, ale také velké množství integrovatelných pluginů a programů usnadňující návrh aplikací (Např. Scene Builder).

Prostředí také zná všechny hlavní používané frameworky a umí upravovat svou nápovědu podle právě používaného frameworku.

Dále nabízí doposud nepřekonané indexování kódu, které umožňuje jednoduchou a rychlou orientaci ve zdrojovém kódu, možnosti refaktorování a velké úpravy kódu.



Obrázek 4.1: Screenshot vývojového prostředí

4.3 Databázový systém

Databáze využívá systém řízení báze dat MySQL, který vlastní firma Oracle Corporation a je dostupný pod bezplatnou licencí GPL (General Public License).

Je to multiplatformní databáze využívající pro komunikaci strukturovaný dotazovací jazyk SQL, která momentálně zabírá velký podíl používaných databází, primárně kvůli snadné implementaci, vysoké bezpečnosti, vysokému výkonu a již výše zmíněné dostupnosti [12]. Jednou z mnoha dalších výhod je možnost používání bez nutnosti internetového připojení, což dokáže značně ulehčit vývoj.

MySQL používá transakce, každé nové klientské připojení obdrží své vlastní vlákno. MySQL také dodržuje ACID zásady databázových transakcí.

4.4 Github

Při vývoji projektu jsem použil verzovací systém Git. Konkrétně jsem použil službu Github.com z důvodu příjemného uživatelského prostředí a jednoduchého ovládání. Důvod použití bylo převážně průběžné zálohování práce, možnost získání starších verzí aplikace a celkové ulehčení správy projektu. Veškeré změny v projektu jsou ukládány a je i případně umožněna distribuce projektu bez nutnosti zbytečného kopírování a posílání zdrojového kódu.

Kapitola 5

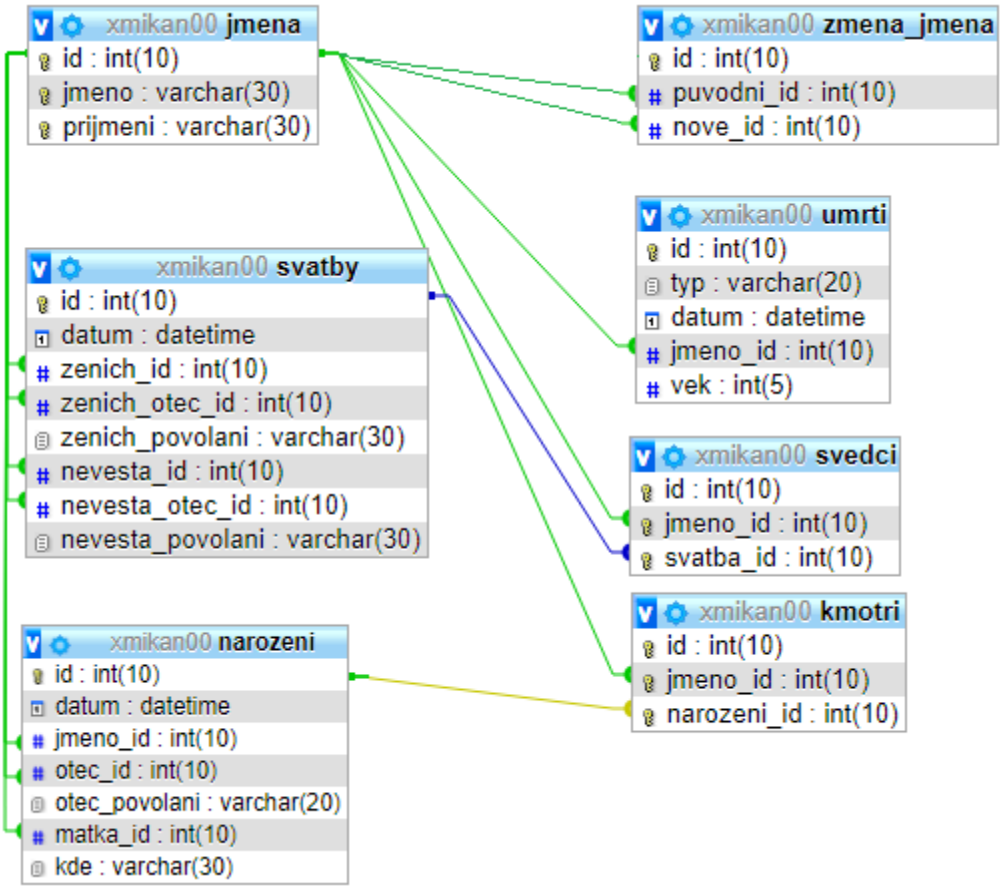
Návrh aplikace

V této kapitole budou rozepsány jednotlivé části systému, funkční komponenty a proces jejich návrhu. Jednotlivé kapitoly budou řazeny v takovém pořadí, aby byla patrná souvislost a proces vytváření aplikace a vztahy mezi jednotlivými částmi systému.

5.1 Návrh databáze

Jako snad první věc, kterou bylo nutné navrhnout, byla databáze. Touto aplikací používaná databáze se skládá z:

- Tabulka **jmena** je používána jako centrální tabulka, která je odkazována jednou, či vícekrát jinými tabulkami a vytváří veškeré vztahy mezi jednotlivými záznamy. Obsahuje id záznamu spolu s příjmením a křestním jménem (prostřední jména jsou ignorována).
- Tabulka **zmena_jmena** obsahuje informaci o změně jména a propojuje tak takové záznamy, které s postupem času měnily jméno (v případě tohoto modelu to jsou pouze manželky).
- Tabulka **umrti** obsahuje typ, datum, věk a odkaz na osobu, o které máme informaci o jejím úmrtí.
- Tabulka **svatby** obsahuje datum svatby spolu s odkazy na ženicha, nevěstu, jejich otce a povolání jejich otců. Dále je k svatbám vázána tabulka **svedci**, která obsahuje odkaz na svědka a svatbu, které je svědkem.
- Poslední tabulka **narození** obsahuje záznam o narozeném člověku a jeho rodičích, datumu a místě narození, také obsahuje informaci o povolání otce. Dále je k narození vázána tabulka **kmotri**, obsahující odkaz na kmotra a narození.



Obrázek 5.1: Databázová struktura

5.2 Zpracování dat

Jako archivní zdroj dat, která jsou použita pro tento projekt, slouží matriky z obcí Jevišovka, Dobré Pole a Nový Přerov od roku 1686 až 1784. Záznamy z matrik byly ručně přepsané do elektronické podoby ve formě excel tabulek.

5.2.1 Nekonzistence a spolehlivost dat

Vzhledem ke způsobu zápisu do matrik za daných let (více mluvených jazyků, jiná definice příjmení) a případné nedokonalosti přepisu, jak kvůli čitelnosti, tak z mnoha dalších důvodů, bylo nutné zpracovávat data s určitou rezervou za cenu případných vztahů mezi záznamy či vůbec správné extrakce samotného jména (viz dále).

5.2.2 Parsování dat

Z poskytnutých dat jsem se rozhodl zakomponovat data ze sešitů svateb, narození a úmrtí, která poskytují většinu potřebných dat pro naše účely. Pro parsování dat jsem napsal parsovací script v jazyce Python, který vezme naše data ve formátu xlsx a uloží je do databáze. Informace ohledně parsování v jazyce python jsem čerpal z [11].

Ze všech informací ze svateb jsem považoval za užitečné datum svatby, ženicha, nevěstu, jejich otce s povoláním a samozřejmě svědky. Ze sešitu úmrtí jsem považoval za důležité datum, typ (důvod), věk a jméno člověka. Z narození datum narození, jméno narozeného, jeho rodiče a místo narození.

Získání jmen

Prvním krokem parsovacího scriptu je extrakce a uložení všech jmen vyskytujících se kdekoliv ve všech sešitech a uložení těchto jmen do tabulky jmena. Každé jméno může být v tabulce pouze jednou.

Svatby V sešitu svateb je vždy dostupné křestní jméno ženicha, rodina ženicha, jméno nevěsty, rodina nevěsty. U většiny záznamů je dále křestní jméno otce ženicha a nevěsty. Dále tabulka obsahuje variabilní počet svědků.

Jednotlivé osoby jsou tedy tvořeny z hodnot:

- Jméno ženicha + jméno rodiny ženicha
- Jméno otce ženicha + jméno rodiny manžela
- Jméno nevěsty + jméno rodiny nevěsty
- Jméno nevěsty + jméno rodiny ženicha
- Jméno otce nevěsty + jméno rodiny nevěsty

Data svědků jsou trochu složitější. Jednotliví svědci jsou odděleni čárkami. Svědky beru v potaz až od 522 zápisu v tabulce, a to kvůli nekonzistentním a nekvalitním datům. Dále všechna jména svědka musejí mít minimálně 3 znaky. V opačném případě jsou považována za neúplné, nebo jako něco jiného než jméno. Také se od 947 záznamu zdánlivě prohodilo pořadí křestního jména a příjmení, což u nějakých později extrahovaných zápisů mohlo způsobit nekonzistenci. Následně jsou odstraněny znaky „!“ a „?“, které předpokládám značily nejistotu přepisujících o validitě či jasnosti zápisu.

Úmrtí V sešitu úmrtí je velmi málo záznamů a jediné jméno z tohoto sešitu je pouze jméno a příjmení člověka, který umřel.

Narození V sešitu narození je vždy dostupné křestní jméno narozeného, příjmení, jméno otce a jméno matky. Křestní jméno, příjmení a jméno otce jsou v dobrém formátu a nevyžadují mnoho úprav. Jména matky jsou však jedno od druhého velmi odlišná. Pokud křestní jméno matky obsahuje dvě a více slov a jedno z těch slov není „Maria“ (např. Anna Maria), považujeme celý záznam jako jméno a příjmení. V případě „Maria“ považujeme celý záznam jako křestní jméno a připojíme příjmení ze sloupce příjmení.

Jednotlivé osoby jsou tedy tvořeny z hodnot:

- Jméno narozeného + příjmení
- Jméno otce + příjmení
- Jméno matky + příjmení (pokud křestní jméno matky obsahuje více slov, pravidla se mění (viz vysvětlení výše)

Data kmotrů jsou velmi nekonzistentně naformátována. Z těchto dat je zaprvé nutné odstranit určité řetězce, které tam jsou pro naše účely navíc. Dále jsou jednotlivá jména rozdělena buďto pomocí čárky, „ a „, nebo „rust“. Za validní jména kmotrů považuji jména obsahující alespoň 3 znaky.

Pravidla pro zpracování kmotrů:

- Pokud má kmotr pouze jedno jméno a žádný zápis v řádku před ním neměl příjmení – ignoruji tento zápis
- Pokud má jedno jméno, ale nějaký zápis ve stejném řádku již měl příjmení, použijeme to
- Pokud má jméno a příjmení – ideální stav
- Pokud jedno jméno obsahuje tři slova – první slovo je považováno za křestní jméno a třetí slovo jako příjmení
- Ostatní stavy jsou ignorovány

Jako poslední krok v získávání jmen jsou odstraněny duplikáty formou přidání unikátního indexu na kombinaci jména a příjmení.

Získání samotných dat

Svatby V sešitu svateb je, kromě výše zmíněných sloupců obsahujících informace o osobách ve formě jmen, ještě informace o rodu (index), datumu svatby, u některých záznamů povolání ženicha a nevěsty a místo, kde obě rodiny žijí. Z těchto dat jsem považoval za relevantní datum svatby a povolání obou nových manželů. Jednotlivé záznamy považuji za validní, pokud obsahují datum, ženicha a nevěstu. Do tabulky svateb jsou uloženy záznamy datum a povolání spolu s odkazy (cizí klíč) do tabulky jmen.

Poté se do tabulky svědci uloží záznamy obsahující primární klíč svatby a příslušného jména.

Úmrtí V sešitu úmrtí jsou, kromě jména zemřelého, informace o člověku, který přijal daného člověka (přepokládám jméno kněze), typ úmrtí (příčina), datum úmrtí, věk, odkud člověk pocházel a kde byl uskutečněn pohřeb. Vzhledem ke stavu dat jsem se rozhodl použít sloupce typ úmrtí, datum a věk (typ a věk nejsou nutné) spolu, samozřejmě, s odkazem na zemřelého do tabulky jmen.

Narození Sešit obsahující informace o záznamech narození, zahrnuje kromě sloupců obsahujících informaci o osobách také datum narození, povolání otce a kde proběhlo narození, s tím, že informace o místě narození není nutná k validitě záznamu. Do tabulky narození jsou uloženy všechny tyto informace.

Následně se samozřejmě uloží informace o kmotrech do tabulky kmotri, obsahující odkaz na narození a příslušné jméno.

Každý kmotr a svědek může být kmotrem či svědkem pouze jednou na každé svatbě, či při každém narození.

Testovací data Na úplný úvod scriptu jsem si také připravil 22 osob mého vlastního rodu, které mají ideální kvalitu záznamu narození a svateb a nad kterými lze příjemně vyvíjet a demonstrovat aplikace.

5.3 Role osob

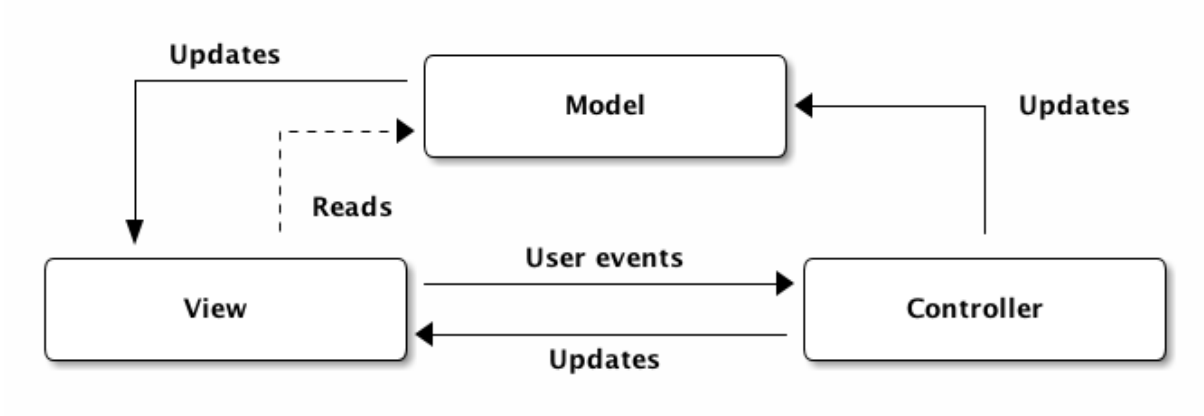
V této části budou vypsány jednotlivé role, kterých mohou objekty nabýt vzhledem k dostupným datům. Dále bude u každé role podmínka, kterou musí daná osoba splňovat, aby mohla mít přiřazenou danou roli. Podle jednotlivým osobám přiřazených rolí může objekt poté figurovat v různých sociálních modelech v různých funkcích.

- otec - osoba může mít přiřazenou roli otce, pokud figurovala jako otec v nějakém záznamu o narození nebo pokud figurovala jako otec ženicha, nebo nevěsty na nějaké svatbě
- matka - osoba může mít přiřazenou roli matky, pokud figurovala jako matka v nějakém záznamu o narození
- potomek - osoba může mít přiřazenou roli potomka, pokud figurovala jako narozený v nějakém záznamu o narození, pokud figurovala jako ženich na nějaké svatbě a pokud figurovala jako nevěsta na nějaké svatbě
- zenich - osoba může mít přiřazenou roli ženicha, pokud figurovala jako ženich na nějaké svatbě
- nevesta - osoba může mít přiřazenou roli nevěsty, pokud figurovala jako nevěsta na nějaké svatbě
- kmotr – osoba může mít přiřazenou roli kmotra, pokud figurovala jako kmotr při nějakém narození
- svedek - osoba může mít přiřazenou roli svědka, pokud figurovala jako svědek při nějaké svatbě
- umrti – osoba může mít přiřazenou roli umrtí, pokud o ní je záznam v tabulce úmrtí

5.4 MVC model

JavaFX umožňuje velmi příjemnou možnost implementace MVC návrhové architektury. Principem MVC architektury je v podstatě oddělení logiky programu od výstupu. M – Model obsahuje ideálně veškerou logiku aplikace a co do ní spadá. V – View převádí data poskytnutá modelem do podoby vhodné pro prezentování uživateli, jedná se o vizualizaci dat uživateli bez logiky zpracování dat (v JavaFX soubory fxml). C – Controller je takový prostředník, který propojuje různé komponenty aplikace. Komunikuje jak s Modelem, tak s View částí systému [8].

Výhodou tohoto modelu je mnohem jednodušší rozšiřování kódu, přidávání nových komponent do aplikace, celkové udržování aplikace a celkově přidání struktury do projektu.



Obrázek 5.2: Schéma MVC modelu, obrázek převzat z [9]

5.5 Návrh grafického uživatelského rozhraní

Tato část se bude soustředit na proces návrhu uživatelského rozhraní aplikace, původní náčrt hlavního okna aplikace, rozdělení prvků a jejich charakteristiku.

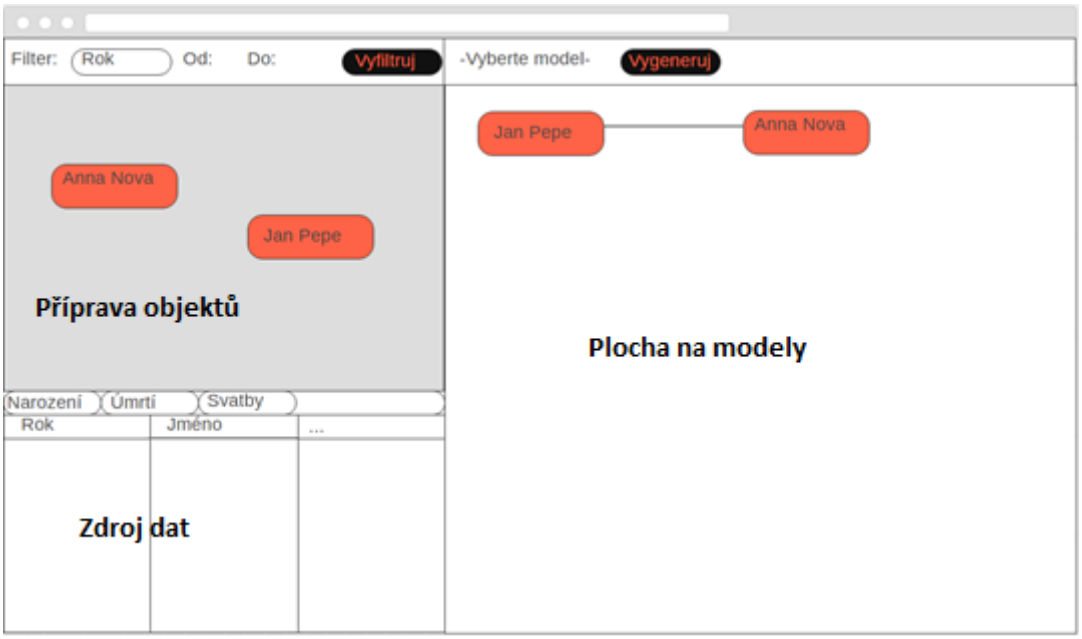
5.5.1 Hlavní okno

Aplikace bude implementována formou jednoho hlavního okna, ve kterém se bude odehrávat většina komunikace s uživatelem.

Při návrhu jednotlivých prvků uživatelského rozhraní jsem se soustředil na některé klíčové prvky návrhu designu [4]:

- Logické (funkčně i vizuálně) rozdělení prvků
- Vzhled prvků by měl odrážet funkci těchto prvků
- Vhodné barevné rozdělení prvků a spojů mezi prvky
- Účel jednotlivých prvků by měl být zřejmý

Vzhledem k tomu, že je aplikace určena pro širokou veřejnost a budou jí využívat lidé nejrůznějšího věku, je aplikace navržena co nejintuitivněji a co nejjednodušeji.



Obrázek 5.3: První návrh uživatelského rozhaní

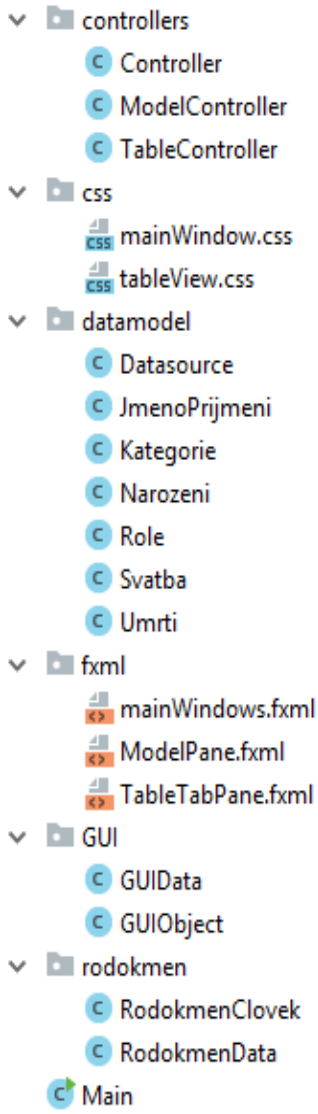
- Zdroj dat
  - Tato plocha slouží jako výpis pro data uložená v databázi ve formátu, který dává smysl. Jednotlivé tabulky jsou rozděleny do tří tabů (narození, úmrtí a svatby), které postupně načítají data z databáze a umožňují přetažení jednotlivých řádků na plochu „přípravy“ objektů.
- Příprava objektů
  - Tato plocha slouží pro přípravu objektů, pro které chceme, aby figurovaly v později vygenerovaných modelech. Jednotlivé objekty se vytvoří pomocí drag and drop operace z některé tabulky plochy „zdroje dat“ a následného vybrání konkrétní osoby. Jednotlivé objekty jsou rozděleny barevně podle jejich známého pohlaví (případně příslušnou neutrální barvou). Pomocí přejetí myší přes tyto objekty zjistíme informace o datumech událostí, ve kterých figurovaly v klíčových rolích, případně jejich povolání. Po kliknutí na určitý objekt se objeví filter v horní části této plochy. Tento filter umožňuje filtrovat existující objekty na této ploše pomocí např. roku, pohlaví a dalších.
- Plocha na modely
  - Tato plocha obsahuje v první části možnost výběru mezi dostupnými modely a tlačítko, které zahájí vytvoření předem vybraného modelu. Zbytek plochy bude použit samotným modelem. Jak bude vygenerovaný model vypadat záleží na konkrétním modelu.

Kapitola 6

Implementace

V této kapitole budou popsány nejdůležitější části vývoje projektu a způsob implementování těchto částí do jednoho celku podle návrhů předchozí kapitoly. Následuje popis vytváření jednotlivých komponent v pořadí, ve kterém vznikaly a problémy, které nastaly v průběhu vývoje, a jejich řešení. Také zde bude rozepsána adresářová struktura projektu, hlavní třídy programu a hlavně dopodrobna rozebraná tvorba sociálních modelů.

6.1 Adresářová struktura projektu



Obrázek 6.1: Adresářová struktura projektu

Všechny soubory projektu jsou rozděleny do jednotlivých skupin podle jejich funkce, typu souboru atd. Adresář *controllers* obsahuje kontroléry projektu. Jednotlivé kontroléry a jejich funkce budou popsány dále v této kapitole.

- Adresář css obsahuje stylovací soubory (soubory které dávají aplikaci její vzhled, jednotlivým prvkům jejich rozměry, a další).
- Adresář *datamodel* obsahuje hlavní třídy projektu včetně třídy Datasource na komunikaci a práci s databází (bude podrobněji popsáno později v této kapitole).
- Adresář *fxml* obsahuje FXML soubory, což jsou soubory napsané v jazyce velmi podobnému XML, který umožňuje popisovat strukturu aplikace a stavět uživatelské prostředí na místě odděleném od aplikační logiky aplikace.
- Adresář *GUI* obsahuje třídy potřebné pro práci s jednotlivými objekty v „přípravě objektů“.
- Adresář *rodokmen* obsahuje třídy potřebné pro vytvoření a modelování rodokmenu.

Dále v rootu projektu je třída, která je vstupní bránou do aplikace.

6.2 Druhy objektů

6.2.1 JmenoPrijmeni

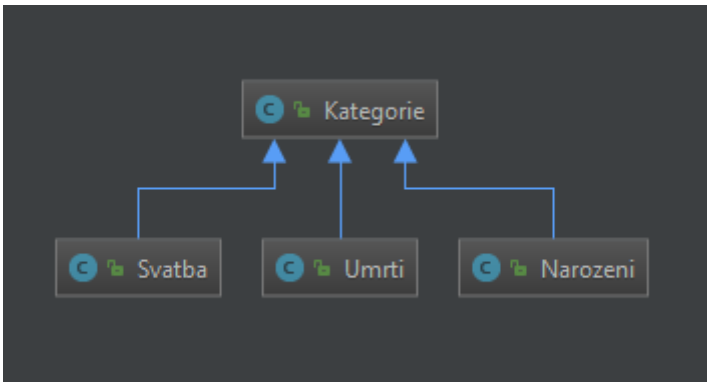
Pro reprezentaci jedné osoby je vytvořena třída JmenoPrijmeni. Jednotlivé tabulky jsou mezi sebou propojeny pomocí cizích klíčů do tabulky osob. Proto je pro správnou manipulaci s daty vytvořena tato třída, která obsahuje atributy *id, jmeno* a *prijmeni*. Dále obsahuje funkci *toString()* , která vrátí vhodnou reprezentaci jména typu String.

6.2.2 Hlavní druhy vztahů

V tomto projektu se, jak již bylo výše zmíněno, pracuje s třemi hlavními typy objektů:

- Narození
- Úmrtí
- Svatba

V programu jsem toto vyřešil pomocí 4 hlavních tříd, kterými jsou reprezentovány tyto objekty. Hlavní rodičovská třída, pojmenovaná *Kategorie*, obsahuje atributy, které jsou společné pro všechny tyto tři typy. Tyto atributy jsou *id* a *datum*. Z této třídy následně dědí třídy *Narozeni, Umrti* a *Svatba*.



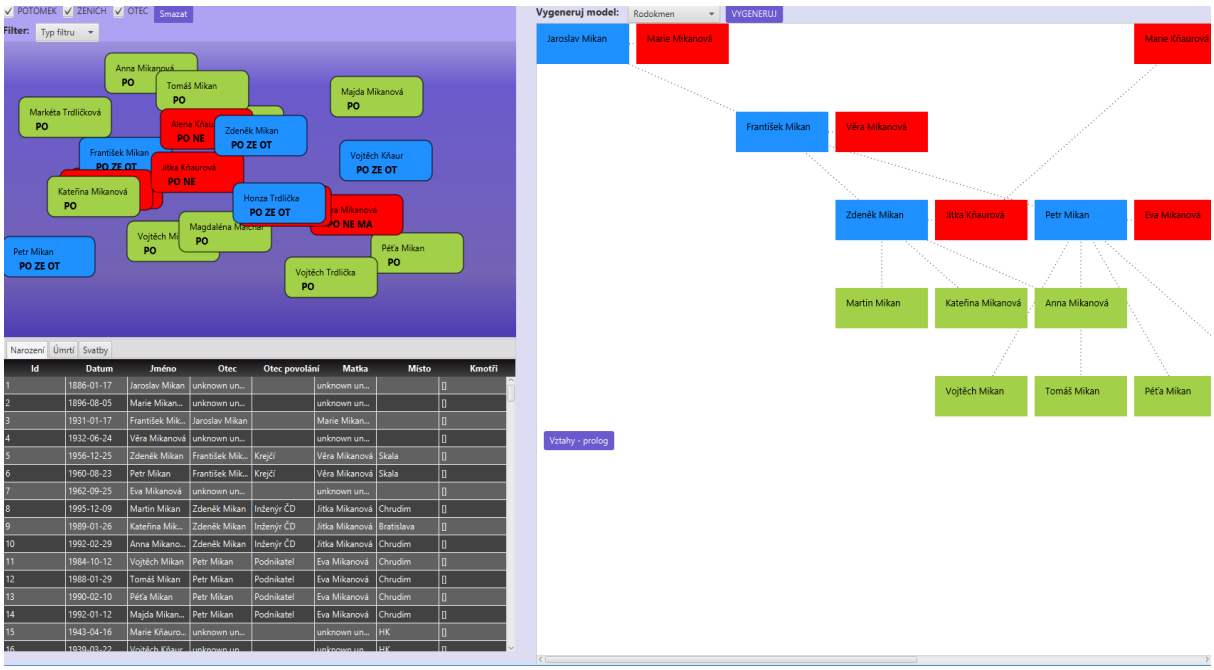
Obrázek 6.2: Diagram dědičnosti hlavních tříd objektů

Narození Narození obsahuje tři atributy typu *JmenoPrijmeni*. Tyto atributy jsou jmeno (narozený), otec a matka. Dále obsahuje dvě *SimpleStringProperty* otecPovolani a kde (místo narození) a ArrayList obsahující objekty typu *JmenoPrijmeni* se jménem kmotri (různě dlouhý seznam kmotrů).

Umrti Úmrtí obsahuje jeden atribut typu *JmenoPrijmeni* – jmeno. Dále obsahuje *SimpleStringProperty* typ (důvod smrti) a *SimpleIntegerProperty* vek (věk v čas smrti).

Svatba Svatba obsahuje 4 atributy typu *JmenoPrijmeni*. Tyto atributy jsou zenich, zenichOtec, nevesta a nevestaOtec. Dále se zde nachází *SimpleStringProperty* zenichPovolani a nevestaPovolani a ArrayList obsahující objekty typu *JmenoPrijmeni* se jménem svedci (různě dlouhý seznam svědků).

6.3 Rozdělení plochy



Obrázek 6.3: Screenshot celkového rozdělení plochy

- *mainWindow.fxml* je hlavní okno aplikace. Tento soubor obsahuje pravou horní část aplikace (navazování typů jednotlivým objektům, filter nad těmito objekty a samotnou plochu „přípravy objektů“). Dále jsou v tomto souboru naincludovány zbývající plochy. Akce nad v tomto souboru popsanou plochou jsou popsány v *Controller* třídě.
- *TableTabPane.fxml* popisuje strukturu tabů a jednotlivých tabulek. Akce nad tímto souborem definovanými prvky je popsána v *TableController* třídě.
- *ModelPane.fxml* popisuje pravou půlku aplikace. V této časti se vybírají jednotlivé modely k vygenerování a nachází se tu samotná plocha na vykreslení těchto modelů. Tato plocha je scrollovací jak horizontálně, tak vertikálně v případě většího množství dat. Akce nad tímto souborem definovanými prvky je popsána v *ModelPane* třídě.

6.4 Spuštění aplikace

Akce provedené při spuštění aplikace jsou, jako v každé Java aplikaci, definovány ve třídě Main v souboru Main.java, který se nachází v rootu projektu. Ještě před načtením projektu se provede připojení k databázi a v případě že se toto spojení úspěšně neprovede, se aplikace ukončí a vypíše chybovou hlášku *"FATAL ERROR: Nelze se připojit k databázi"*.

6.4.1 Připojení k databázi

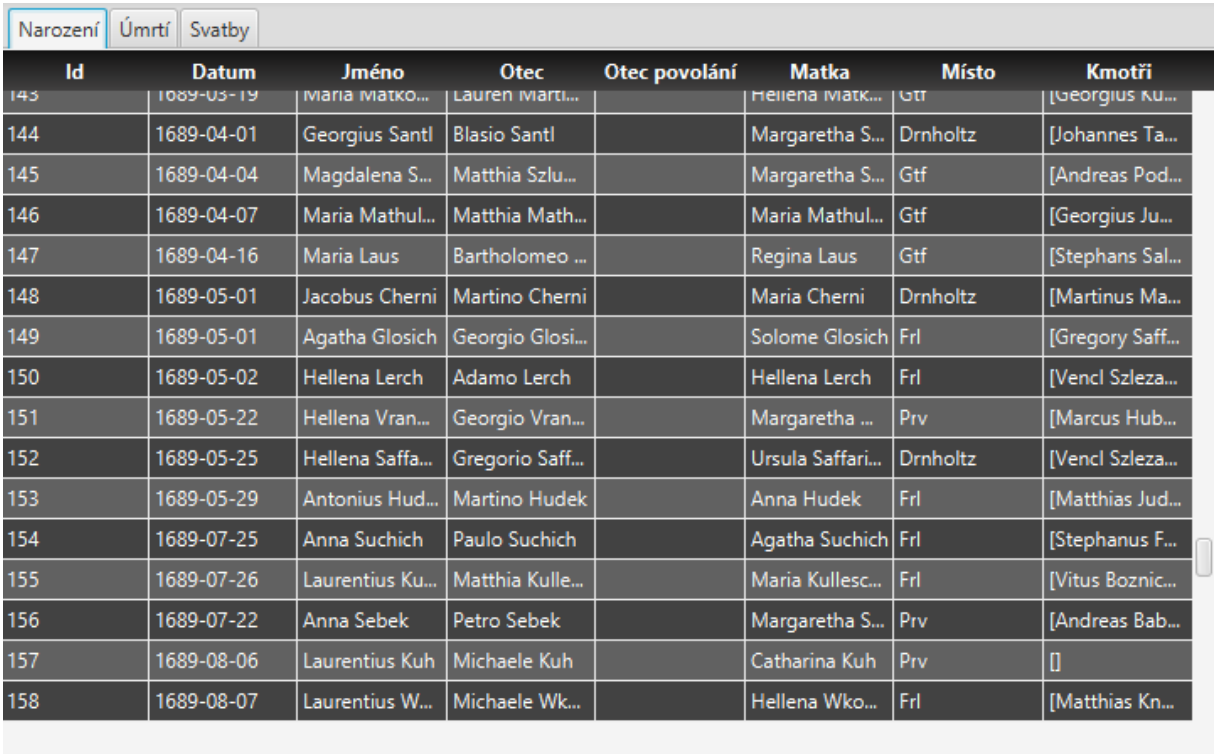
O připojení k databázi, ukončení připojení a veškeré akce čtení / zápisu databáze + nějakou základní práci nad těmito daty se stará třída *Datasource*. V této třídě jsou definovány konstanty potřebné pro připojení k databázi, všechny SQL dotazy do databáze, které jsou později při běhu aplikace využívány, a funkce, které je využívají. Je realizována pomocí vzoru singleton. Při spouštění programu je vytvořena instance třídy, která je pouze jedna pro celý program. K této instanci se poté přistupuje pomocí funkce *getInstance()* odkudkoli v programu.

Při spuštění projektu se zavolá funkce *open()* této třídy, ve které se ustanoví spojení s, v tomto případě, MYSQL serverem a zároveň se všechny SQL dotazy předpřipraví (připojení k databázi umožňuje připravit jednotlivé dotazy, to znamená, že se dotaz pošle předem na MYSQL server na předkompilaci a poté, pokaždé, kdy je dotaz zavolán, je jeho provedení rychlejší a efektivnější). Aplikace počítá s databází se jménem "xmikan00", přihlašovacím jménem "root"a prázdným heslem. V případě, že toto připojení není úspěšné, aplikace končí s informační hláškou.

6.4.2 Samotné spuštění aplikace

Po navázání spojení s databází se provede samotné spuštění aplikace, nastaví se velikost scény (okna aplikace), název tohoto okna a načte se struktura aplikace definovaná v souboru *mainWindows.fxml* a akce ve funkci *initialize()* v kontrolérech spojených s tímto fxml souborem.

6.5 Zdroj dat



Id	Datum	Jméno	Otec	Otec povolání	Matka	Místo	Kmotři
144	1689-04-01	Georgius Santl	Blasio Santl		Margaretha S...	Drnholtz	[Johannes Ta...
145	1689-04-04	Magdalena S...	Matthia Szlu...		Margaretha S...	Gtf	[Andreas Pod...
146	1689-04-07	Maria Mathul...	Matthia Math...		Maria Mathul...	Gtf	[Georgius Ju...
147	1689-04-16	Maria Laus	Bartholomeo ...		Regina Laus	Gtf	[Stephans Sal...
148	1689-05-01	Jacobus Cherni	Martino Cherni		Maria Cherni	Drnholtz	[Martinus Ma...
149	1689-05-01	Agatha Glosich	Georgio Glosi...		Solome Glosich	Frl	[Gregory Saff...
150	1689-05-02	Hellena Lerch	Adamo Lerch		Hellena Lerch	Frl	[Vencl Szleza...
151	1689-05-22	Hellena Vran...	Georgio Vran...		Margaretha ...	Prv	[Marcus Hub...
152	1689-05-25	Hellena Saffa...	Gregorio Saff...		Ursula Saffari...	Drnholtz	[Vencl Szleza...
153	1689-05-29	Antonius Hud...	Martino Hudek		Anna Hudek	Frl	[Matthias Jud...
154	1689-07-25	Anna Suchich	Paulo Suchich		Agatha Suchich	Frl	[Stephanus F...
155	1689-07-26	Laurentius Ku...	Matthia Kulle...		Maria Kullesc...	Frl	[Vitus Boznic...
156	1689-07-22	Anna Sebek	Petro Sebek		Margaretha S...	Prv	[Andreas Bab...
157	1689-08-06	Laurentius Kuh	Michaele Kuh		Catharina Kuh	Prv	[]
158	1689-08-07	Laurentius W...	Michaele Wk...		Hellena Wko...	Frl	[Matthias Kn...

Obrázek 6.4: Screenshot plochy zdroje dat

Tabulka zdroje dat se nachází v levém dolním rohu aplikace. Vzhled a struktura této tabulky je popsána v souboru *TableTabPane.fxml.* Tento soubor obsahuje TabPane (GUI prvek, který umožňuje měnit mezi jednotlivými taby – viz horní část této plochy).

Dále pro každý jednotlivý tab (Narození, Úmrtí, Svatby) obsahuje popsání jednotlivých sloupců tabulky s jejich popisem a zároveň pro každý tab existuje progress bar, který zobrazuje proces načítání dat z databáze do tabulky (databáze obsahuje velké množství dat a trvá určitou dobu, než se všechna data načtou – pro každou tabulku jiné). Akce nad tabulkou jsou definovány v kontroléru *Tablecontroller.*

6.5.1 Naplnění tabulek daty

Naplnění tabulek je synchronizováno. Souběžnost je v Javě realizována pomocí vláken (Thread). Předtím, než můžeme spustit realizování nového vlákna, musíme však definovat jeho akci. Tento kód se nazývá Task (úkol), pro naše potřeby je nutné vytvořit tři nové úkoly, vždy na získání všech dat tabulky z databáze. Úkoly jsou pojmenovány *GetAllUmrtiTask, GetAllSvatbyTask* a *GetAllNarozeniTask.* Tyto třídy rozšiřují java třídu Task a je v nich přepsána funkce *call()*, která je volána při spuštění vlákna.

V této funkci se pro každou tabulku zavolá příslušná funkce třídy *Datasource*, která zavolá SQL dotaz nad databází, který vrátí všechny sloupce příslušné tabulky. Všechny tyto sloupce jsou následně přetypovány na jejich cílený typ a přiřazeny objektu jejich konkrétní třídy.

V databázi jsou osoby uloženy pomocí cizího klíče do centrální tabulky osob. Proto je při získávání dat z databáze ještě nutné pro sloupce, obsahující právě toto ID, provést další dotaz nad databází pro získání jména a příjmení pro konkrétní ID a vytvoření objektu třídy *JmenoPrijmeni* s těmito daty, které je poté přiřazeno instanci naší třídy.

Jednotlivé úkoly vracejí *ObservableList*, ve kterých jsou objekty buďto Umrti, Svatba nebo Narozeni. List typu *ObservableList* je zde využitý z důvodu možnosti přidání posluchačů (akcí reagujících na změnu) nad tabulkou a nastavení námi vyžadovaného chování jako reakci.

Před spuštěním úkolu je také nastavena viditelnost ukazatele průběhu, který je vždy zobrazen pod příslušnou tabulkou a značí nám, že všechna data ještě nebyla načtena z databáze. V případě že úkol uspěje, i v případě že z jakéhokoliv důvodu selže (ztráta spojení s databází, ...), ukazatel průběhu bude zneviditelněn.

Executor Service Jednotlivé úkoly jsou spuštěny a spravovány pomocí takzvané *ExecutorService*. Executor service je náhrada pro přímou práci s vlákny. Exekutory jsou schopné spouštět více vláken najednou a také udržovat použitá vlákna pro budoucí použití, proto si vystačíme pouze s jednou instancí tohoto objektu pro celý běh aplikace.

Jednotlivé úkoly jsou po definování jejich kódů poslány na zpracování exekutoru a aplikace se o ně dále nemusí starat a posouvá se dále na další úkoly.

Po poslání posledního úkolu na zpracování exekutorem exekutor čeká 90 sekund na ukončení všech jeho vláken a poté se ukončí, u čeho případně přeruší zpracovávání všech vláken, která jsou stále aktivní. Tato situace může nastat při nějaké neočekávané chybě v průběhu zpracovávání jednotlivých úkolů.

6.5.2 Drag and Drop

První akcí při inicializování tabulek je přidání možností tahání jednotlivých řádků (drag and drop) na všechny řádky tabulek.

Row Factory Tato funkcionalita se dá realizovat pomocí *setRowFactory()* metodou, kterou obsahuje prvek JavaFX TableView, který je právě kvůli této funkcionalitě použit v tomto projektu. RowFactory umožňuje definovat akci nad pouze jedním řádkem, avšak toto nastavení bude replikováno na každý řádek této tabulky, ať bude přidán/odebrán kdykoliv za běhu aplikacen.

Definování akce Drag and Drop Nad tabulkou samotnou, nás zajímá pouze začátek drag and drop akce (tabulka je zdrojová plocha, cílová plocha je plocha přípravy objektů v pravém horním rohu). Nad každým řádkem je tedy pomocí dříve zmíněné row factory přidán posluchač na akci *dragDetected* (klik a držení levého tlačítka myši nad řádkem). Po tomto kliknutí se, v případě, že řádek není prázdný, uloží index řádku a vytvoří se objekt *Dragboard* (nastaví vizuální reprezentaci dat, která jsou přenášena pomocí drag and drop – nastavení kurzoru nad tabulkou a plochou „přípravy objektů“, která značí že probíhá drag and drop)**.** Následně se vytvoří objekt typu Kategorie (pracujeme nad buďto narozením, úmrtím nebo svatbou a tyto třídy dědí z třídy Kategorie), kterému přiřadíme daný objekt.

Abychom mohli pomocí *Dragboard* přenést nějaká data, musíme nejdříve vytvořit „schránku“ pro tato data. K tomuto účelu slouží objekt *ClipboardContent. ClipboardContent* je dočasné místo, do kterého můžeme uložit data primárně k účelu jejich přenesení je mezi různými místy v programu. Data, která budeme pomocí naší drag and drop akce přenášet, budou

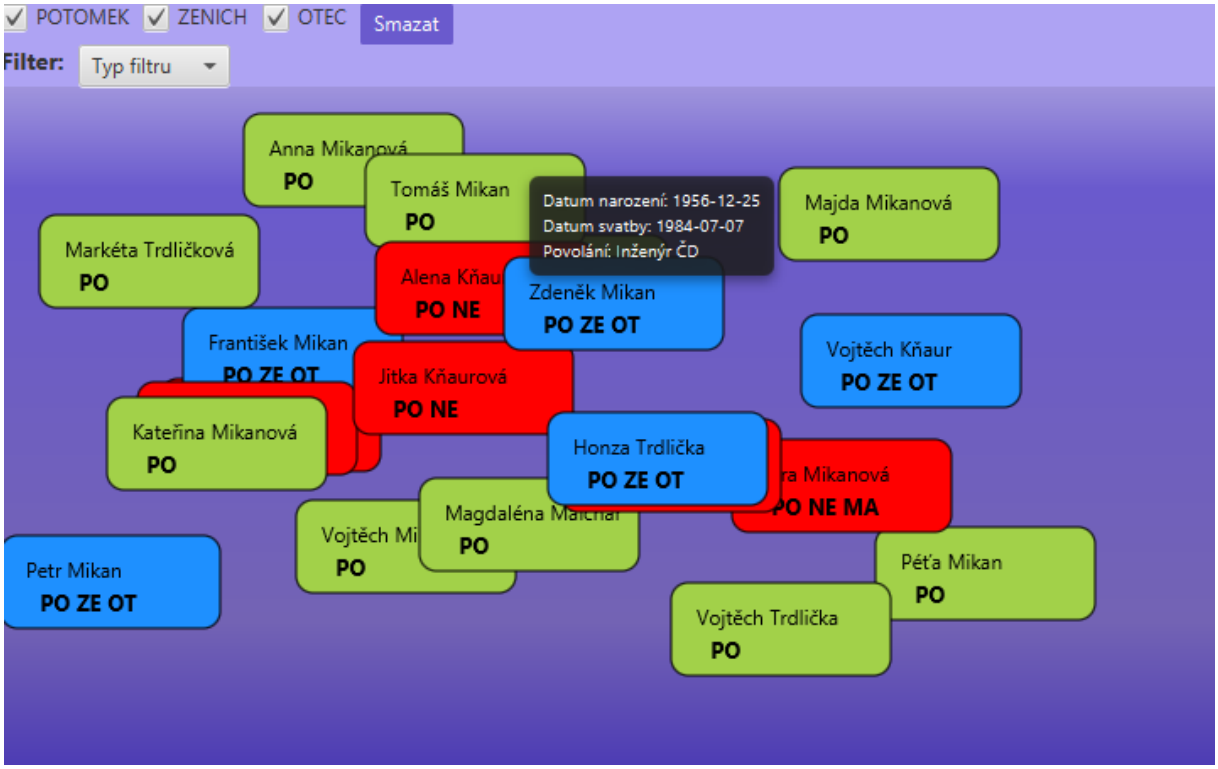
zaprvé *Integer* index a *String*, který nám jednoznačně určí o jaký řádek / objekt se jedná. Tento *String* se skládá ze jména třídy, které je tento objekt instancí (buďto narození, úmrtí nebo svatba) a id tohoto objektu ve formátu „NÁZEV_TŘÍDY + „ „ + ID“.

Data musí být pro naplnění clipboardy nějakým způsobem serializována. Pro tento účel byl použit MIME typ "application/x-java-serialized-object". Poté, co jsou data serializována, jsou uložena do cliboard obsahu, který je následně nastaven jako obsah dříve vytvořeného Dragboard objektu.

6.6 Příprava objektů

Plocha „přípravy objektů“ se nachází v levé horní části aplikace. Vzhled a struktura této tabulky je popsána v hlavním souboru *mainWindows.fxml*. Tuto část GUI nemělo smysl, podle mého názoru, přesouvat do vlastního souboru a vytvářet vlastní controller pro její ovládání, značně by to zkomplikovalo a znepřehlednilo celkový návrh této aplikace a nepřineslo by to žádné relevantní výhody.

6.6.1 Plocha přípravy objektů



Obrázek 6.5: Screenshot plochy přípravy objektů

Tato plocha je tvořena jedním BorderPanem. Toto uspořádání umožňuje rozdělení plochy na horní, pravou a levou, spodní a center části. V horní části se nachází „typeBox“. Je to horizontální box, který obsahuje checkboxy na přiřazení typů objektu (bude probráno do detailu později v této kapitole). V centru tohoto borderPanu se nachází další borderPane s id „mainWindow“. MainWindow je dále rozděleno na horní a center část. V horní části tohoto vnořeného borderPanu se nachází další horizontální box, tentokrát obsahující prostředky pro filtrování existujících objektů (bude taktéž detailně probráno dále v této kapitole). V centru mainWindow se již nachází samotná plocha pro generování objektů.

6.6.2 Způsob uložení připravovaných objektů

Objekty, které se nacházejí v přípravě objektů, jsou ukládány do třídy vzoru singleton *GUIData*. Tato třída obsahuje instanci na sebe sama (singleton návrhový vzor), seznam objektů, které se tam nachází, typu *GUIObject* pojmenovaný GUIObjects a také seznam objektů typu *GUIObject* GuiObjectsFiltered, který obsahuje aktuální seznam relevantních objektů, co se týče filtrování (více ohledně filtrování dále v této kapitole).

GuiObject *GUIObject* třída reprezentuje objekt, který se nachází v přípravě objektů. Obsahuje v první řadě odkaz na danou osobu typu *JmenoPrijmeni*, dále datumy akcí, o kterých víme, že se daný člověk účastnil (datum narození, úmrtí a svatby, ve které figuroval jako ženich / nevěsta). Také může obsahovat informaci o povolání člověka, pokud je nám tato informace k dispozici, a v neposlední řadě seznam rolí, ve kterých člověk někdy figuroval.

Jak již název napovídá, třída také obsahuje grafické prvky. Těmito prvky jsou obdelník, text, který zobrazuje jméno dané osoby, a text rolí, který vyjadřuje, jaké všechny role má osoba v tento moment přiřazené.

6.6.3 Dokončení Drag and Drop akce

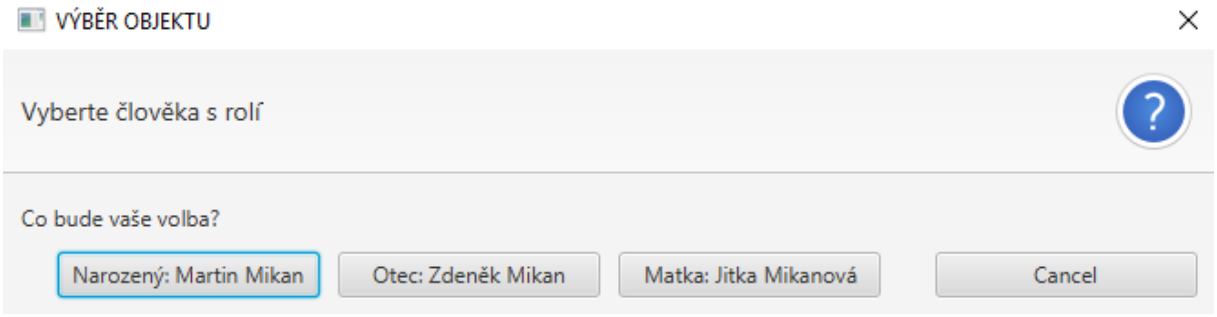
Objekty se do přípravy objektů dostávají pomocí výše zmíněné drag and drop akce z jedné z tabulek dat. Plocha přípravy objektů je plocha, na které se dané objekty přenášejí a která slouží jako konečná plocha tohoto přenosu.

Nad mainWindow jsou přidány dva drag and drop posluchače. První je *OnDragOver*, který změní kurzor, aby bylo jasně čitelné, že toto je ta plocha, na kterou chceme, aby byly objekty přenášeny. Druhý je *OnDragDropped*, který, jak již název napovídá, reaguje na uvolnění levého tlačítka myši, která je v akci drag and drop nad touto plochou.

OnDragDropped Zpracování DragEventu nad mainWindow začíná načtením přenášeného Dragboard objektu. Následně načteme obsah tohoto objektu a deserializujeme ho. Dále rozdělíme přenášený textový řetězec podle bílých znaků tak, že získáme třídu přenášeného řádku tabulky a ID tohoto objektu.

Inicializace objektu Po získání třídy a id objektu je využita třída Datasource na získání konkrétního objektu z databáze. Na sestavení dotazu je zde použit StringBuilder (přehledná konkatenace a tvoření dotazu). Vložíme správnou tabulku do FROM části dotazu a získané ID objektu do WHERE části dotazu. Následně vytvoříme instanci objektu konkrétní třídy a inicializujeme ho výsledkem dotazu.

Zpracování dialogu a inicializace objektu Pro vytvoření konkrétního člověka z objektu narození, úmrtí nebo svatby, kterému lze přiřazovat role a nad kterým je možné vytváření objektů, se zde využívá dialog. Po získání instance konkrétní třídy a naplnění této instance správnými daty se zobrazí formulář, který vyzívá k vybrání konkrétní osoby z tohoto vztahu.



Obrázek 6.6: Screenshot dialogu na výběr objektu

Obsah tohoto dialogu (tlačítka – možné osoby) je tvořen v závislosti na konkrétní třídu. Všechny třídy, které dědí z třídy Kategorie, implementují metodu *getButtonTypes()*, která vrací seznam tlačítek pro všechny jejich *JmenoPrijmeni* objekty.

Osoby v

- Narozeni – narozený, otec, matka a libovolný počet kmotrů
- Umrti – zesnulý
- Svatba – ženich, nevěsta, otec ženicha, otec nevěsty a libovolný počet svědků

Dále všechny třídy, které dědí z třídy Kategorie, implementují metodu *handleButtonAction()*, která zpracovává kliknutí na určité tlačítko a vytvoří *JmenoPrijmeni* objekt podle kliknutého tlačítka. Po vytvoření tohoto objektu je nutné naplnit ho daty z databáze. Pomocí ID se získá z tabulek narození, ve kterém figuruje tento člověk jako narozený, datum jeho narození (předpokládá se pouze jeden existující záznam), dále datum úmrtí z tabulky úmrtí a datum svatby, ve kterém figuruje jako ženich, či nevěsta.

Další atribut, který GUIObject obsahuje, je povolání člověka. Tato informace se nachází buďto v záznamu narození, kde osoba figuruje jako otec, nebo v záznamu svatby, pokud figuruje jako ženich nebo nevěsta.

Možné role osob Další důležitý krok při inicializaci objektu je zjištění všech možných rolí, které mohou být objektu přiřazeny. Prerekvizity pro přiřazení role objektu jsou popsány v kapitole Návrh – role osob. Všechny možné role, kterých osoby mohou nabývat, jsou popsány ve třídě *Role*. V této třídě se nachází výčtový typ pro jednotlivé role se jménem *ROLES*. Výčtový typ nám ulehčuje přiřazování a porovnávání hodnot a je vždy soukromý a finální. Jen pro připomenutí, tyto role jsou OTEC, MATKA, POTOMEK, ZENICH, NEVESTA, KMOTR, SVEDEK a UMRTI. Pro každou osobu se prochází všechny možné vztahy, ve kterých figuroval, a postupně se určí všechny role, kterých kdy člověk nabyl. Tyto role jsou následně uloženy do řetězce typu *Role* GUIObjektu. Výchozí stav přiřazených rolí k objektu je, že všechny možné získané role jsou přiřazeny.

Nastavení grafiky GUIObjektu Po kompletním inicializování všech datumů, povolání a všech možných rolí zbývá již pouze nastavit všechny grafické prvky GUIObjektu.

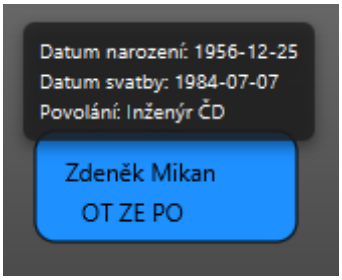
Obdelník Prvním objektem je obdélník, graficky reprezentující objekt samotný. Šířka a výška obdélníku je pevně daná, dostatečně velká na udržení textu, který přes tento obdélník bude vypsán. Koordinace obdélníku jsou převzaté z drag and drop eventu. V případě, že je objekt přetažený na takové místo, které neumožňuje jeho vykreslení okolo tohoto bodu, je objekt dostatečně posunut od kraje. Obdélníku se dále nastavuje barva pozadí podle pohlaví osoby, kterou reprezentuje. Pokud je jedna z možných rolí osoby otec nebo ženich, je obdélník s modrým pozadím. Pokud je jedna z možných rolí matka nebo nevěsta, je pozadí červené. V případě, že osoba nefigurovala jako ani jedna z těchto rolí, je barva neutrální světle zelená.

Texty Zbylé grafické prvky jsou texty jména a rolí. Text jména pouze vezme jméno osoby a zobrazí ho v horní části obdélníku ve formátu JMÉNO + „ „ + PŘIJMENÍ. Informace o rolích, které má objekt právě přiřazené, je zřejmé podle textu rolí. Tento text se nachází ve spodní části obdélníku. Jednotlivé role jsou vypsány tak, že jsou vzaty první dvě písmena z dané role a jsou vypsány velkým písmem jedna role po druhé.

Na oba texty je také navázán eventHandler, který přesune všechny události způsobené myší nad těmito texty na obdélník.

Informační bublina Všechny informace, které známe o osobě (datum narození, svatby, úmrtí a povolání), je možné získat přejetím myši přes obdélník. Pokud některé tyto informace o osobě známe, vytvoří se objekt typu Tooltip, který vypíše tyto informace ve formátu:

- "Datum narození: "DATUM
- "Datum svatby: "DATUM
- "Datum úmrtí: "DATUM
- "Povolání: "POVOLÁNÍ



Obrázek 6.7: Screenshot thumbnailu objektu

Přetahování objektu Další vlastností grafických objektů v bloku přípravy objektů je možnost je přetahovat na různé pozice, což nám umožňuje si vizuálně uspořádat objekty podle našich potřeb. Tato funkcionalita je dosažena pomocí *setOnMousePressed* a *setOnMouseDragged*, kdy kliknutí myši posune kliknutý grafický shluk (obdelník a texty) do popředí a uloží si do atributů třídy pozici kliknutí. Při posouvání objektu se vypočítá nová pozice podle pozice myši a začáteční pozice kliku, kterou jsme si uložili při započetí pohybu. Pokud se snažíme přetáhnout daný shluk mimo plochu přípravy objektů, shluk se nám zastaví na hranici této plochy.

6.6.4 Test data

Uprostřed této plochy je také pro testovací účely umístěno tlačítko, které vloží 22 objektů mého rodu do přípravy objektů. Tyto objekty mají mezi sebou ideální vazby (nechybí žádné informace a jména jsou korektního formátu) a jsou užitečné jak pro vývoj, tak pro demonstraci funkcionality aplikace.

6.6.5 Odstranění objektů

Příprava objektů by nebyla kompletní bez možnosti smazat již existující objekty – tato funkcionalita je dosažena pomocí typeBox horizontálního boxu v horní části přípravy objektů. Tato plocha je zobrazena po kliknutí na určitou osobu (obdélník). Po kliknutí se zobrazí plocha na přiřazování rolí (viz dále), na které je tlačítko, které smaže objekt z plochy přípravy objektů a ze seznamu filtrovaných objektů.

6.6.6 Přiřazování rolí

Obrázek 6.8: Screenshot plochy na přiřazování rolí

Důležitou částí přípravy objektů na modelování je přiřazení rolí, které budou brány v potaz při modelování, jednotlivým osobám. Po kliknutí na grafický prvek se projdou všechny možné role pro danou osobu a pro každou roli se vytvoří zaškrtávací pole, které je zaškrtnuté, pokud je daná role objektu právě přiřazena. Na všechna políčka je také navázán posluchač na změnu stavu políčka. Pokud je role nově přiřazena, přidá se role do seznamu přiřazených rolí a přepíše se text rolí u daného objektu, pokud je role odstraněna, odstraní se role ze seznamu přiřazených rolí a přepíše se text rolí.

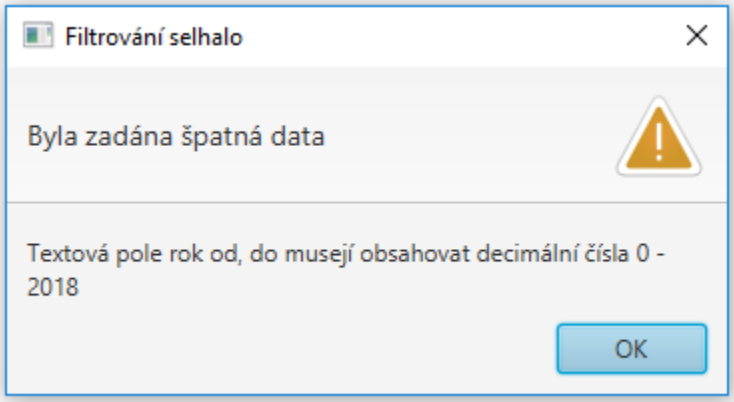
6.6.7 Filtrování objektů

Obrázek 6.9: Screenshot plochy na filtrování podle roku

Další neméně důležitá součást přípravy objektů je filtrování již vytvořených objektů. Momentálně lze filtrovat pomocí roku či podle pohlaví. V horní části plochy přípravy objektů se nachází ComboBox, který nám umožňuje výběr jednoho z těchto filtrů. Před každým filtrováním je seznam GuiObjectsFiltered vyprázdněn a znovu naplněn objekty, které úspěšně prošly filtrem.

Filtrování podle roku Po vybrání filtrování podle roku z výše zmíněného combo boxu se zobrazí formulář s políčky od a do a se zaškrtávacím políčkem „ponechej neznámé“. Textová pole očekávají číselný hexadecimální vstup v rozsahu 0 – současný rok. V případě, že alespoň jeden z roků je menší nežli 0, větší nežli současný rok, nebo je rok „od“ větší, než

rok „do“, zobrazí se chybová hláška se stručnou nápovědou na používání filtru. V případě, že je jedno z políček vynechané, vyplní se automaticky krajní hodnotou.



Obrázek 6.10: Screenshot alertu po špatných datech ve filtru

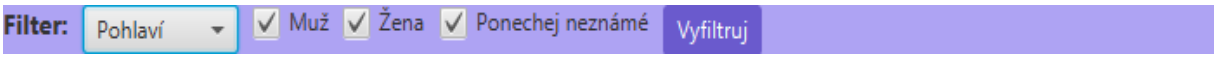
Zaškrtávací políčko „ponechej neznámé“ nám, v případě, že je zaškrtnuté, ponechá objekty, u kterých nemáme žádnou informaci typu datum (narození, svatby, či úmrtí). V opačném případě všechny tyto objekty vyfiltruje.

V případě, že uživatel vyplnil formulář správnými daty, filtr začne postupně procházet všechny objekty, které se v tento moment nachází v případě objektů (pokud jsou nějaké objekty momentálně vyfiltrované, nová akce filtrování bude procházet i tyto objekty – nelze na sebe skládat filtry). Pro každý jednotlivý objekt získáme jednotlivé roky událostí, které jsou osobě přiřazené pomocí java třídy Calendar. Po získání všech těchto hodnot postupně porovnáme, zda-li je datum narození větší, nebo rovno hodnotě „od“, a menší nebo rovno hodnotě „do“. Pokud tato situace nastane, znovu tento objekt vykreslíme a přidáme do seznamu GuiObjectsFiltered (pokud již byl vykreslen, pokračujeme dalším objektem). Pokud tato situace nenastane, pokračujeme stejným způsobem s rokem úmrtí a rokem svatby. Pokud bylo zaškrtnuto políčko „ponechej neznámé“ a nemáme žádnou informaci, provede se stejná akce.

V případě, že není zaškrtnuto políčko „ponechej neznámé“ a osoba nemá přiřazené žádné datumy, nebo když ani jeden z datumů nesedí danému rozsahu, grafické prvky spojené s touto osobou budou odstraněny z plochy „přípravy objektů“ a GUIObject.

Filtrování podle pohlaví Při vybrání filtrování podle pohlaví se zobrazí zaškrtávací políčka „Muž“, „Žena“ a „Ponechej neznámé“. Po odeslání formuláře se postupně začnou procházet všechny objekty, které se v tento moment nachází v případě objektů (pokud jsou nějaké objekty momentálně vyfiltrované, nová akce filtrování bude procházet i tyto objekty – nelze na sebe skládat filtry). Pro každý objekt získáme jeho pohlaví stejným způsobem, jako byla získána jeho barva – pokud je jedna z možných rolí osoby otec nebo ženich, předpokládáme mužské pohlaví, pokud je jedna z možných rolí matka nebo nevěsta, předpokládáme ženské pohlaví. V případě, že osoba nefigurovala jako ani jedna z těchto rolí, neznáme pohlaví osoby.

V případě, že je zaškrtnuté políčko „Muž“, a osoba figurovala ve výše zmíněných rolích, znovu tento objekt vykreslíme a přidáme do seznamu GuiObjectsFiltered (pokud již byl vykreslen, pokračujeme dalším objektem). Pokud tato situace nenastane, pokračujeme stejným způsobem s ženským pohlavím a s neznámým pohlavím osoby. Pokud pohlaví osoby nesouhlasí s pohlavím vyplněným ve formuláři, grafické prvky spojené s touto osobou budou odstraněny z plochy „přípravy objektů“ a GUIObject.



Obrázek 6.11: Screenshot plochy na filtrování podle pohlaví

6.7 Generování modelů

Obrázek 6.12: Screenshot plochy na výběr modelu na vygenerování

Plocha „generování modelů“ se nachází v pravé polovině aplikace. Vzhled a struktura této části je popsána v souboru *modelPane.fxml* a její logika je popsána v příslušném kontroléru *ModelController*. Celá tato plocha je jeden vleký BorderPane, který je rozdělen na horní a centrální části. V horní části se nachází formulář na výběr typu modelu, který chceme vygenerovat, spolu s tlačítkem na samotné vygenerování. Zbytek plochy je vyplněn ScrollPanem – prvek v JaveFX, který umožňuje procházet obsah v tomto prvku pomocí horizontálních a vertikálních posuvníků.

6.7.1 Příprava plochy

Výška a šířka plochy na vykreslení modelu je fixně daná (950 na 1200 pixelů), každopádně velikost plochy samotné se odvíjí od vykreslovaného modelu. Horizontální i vertikální posuvník je vždy zobrazen (hned je zřejmé, že existuje tato funkcionalita).

Plocha na kreslení modelů není tvořena samotným scrollPanem. Uvnitř objektu ScrollPane se vygeneruje obdélník bílé barvy, podle jehož velikosti se konfigurují posuvníky a na kterém se vyobrazují jednotlivé grafické prvky vykreslovaných modelů.

6.7.2 Generování modelů

Generování modelů začíná zpracováním formuláře. Podle vybraného modelu k vygenerování se zavolá patřičná rutina zpracování, která zpracuje objekty uložené v seznamu GuiObjectsFiltered.

Rozdělení osob podle pohlaví První z modelů, které jsou dostupné k vygenerování, je rozdělení vybraných osob do skupin podle pohlaví. Tento model přehledně vypíše všechny

osoby, které jsou inicializované v přípravě objektů a vykreslí je. Umožňuje si projít všechny připravované objekty – najít objekty, které tam případně nepatří a jasně si ustanovit, nad kterými osobami vlastně pracujeme.

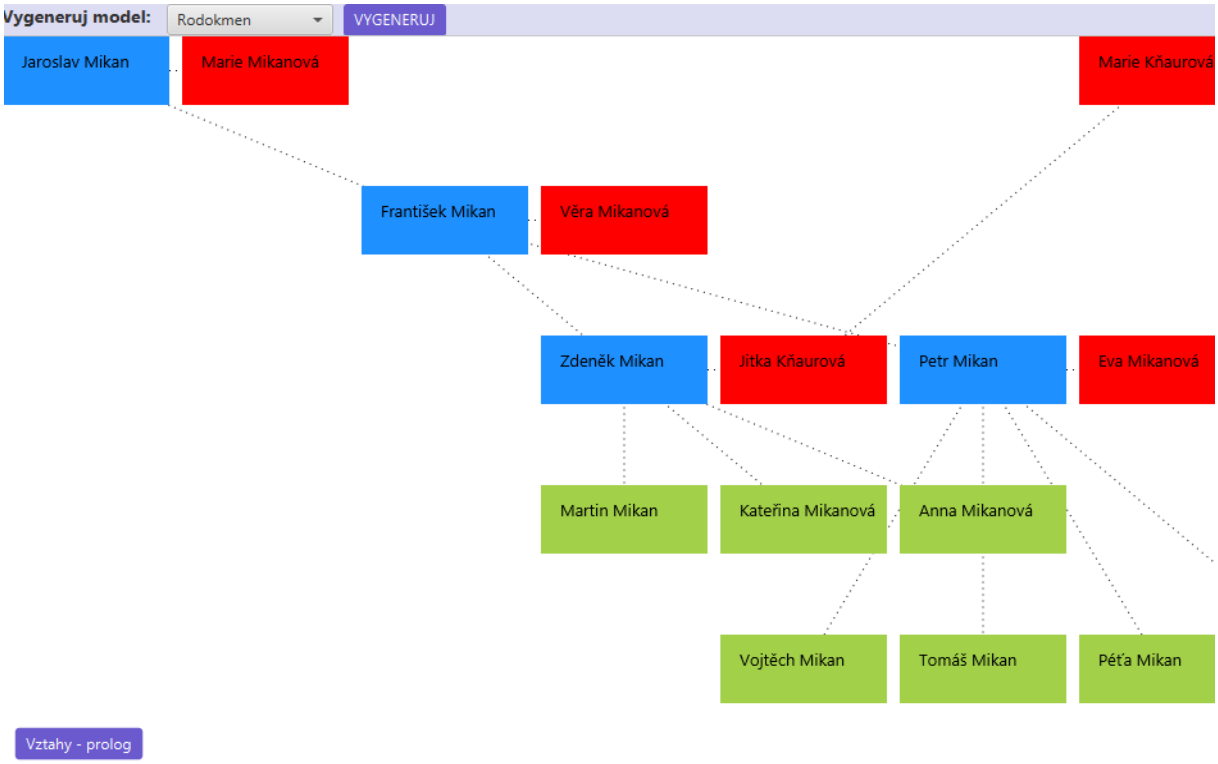
Obrázek 6.13: Screenshot modelu rozdělení osob podle pohlaví

Vykreslení modelu je implementováno pomocí postupného procházení všech objektů, které se nachází se v seznamu GuiObjectsFiltered pro každé pohlaví. Jak již bylo dříve ustanoveno, pokud je jedna z možných rolí osoby otec nebo ženich, předpokládáme mužské pohlaví, pokud je jedna z možných rolí matka nebo nevěsta, předpokládáme ženské pohlaví. V případě, že osoba nefigurovala jako ani jedna z těchto rolí, neznáme pohlaví osoby.

První skupina osob, která je hledána, jsou muži. Postupně se prochází seznam osob a v případě, že byla nalezena osoba, u které předpokládáme mužské pohlaví, je vykreslen obdélník modré barvy, nad kterým je vypsáno jméno příslušné osoby ve fontu „Open Sans“. Všechny další obdélníky jsou vykreslovány vedle sebe, dokud nedosáhneme konce plochy. V tomto případě se další objekty začnou vykreslovat na další řádek. Důvod, proč se vykreslují jednotlivé řádky pod sebe, místo jedné konečné řady obdélníků, je primárně přehlednost výsledného modelu.

V případě, že byla nalezena osoba, u které předpokládáme mužské pohlaví, posuneme souřadnici y o dvě řady a začneme vykreslovat objekty pro další pohlaví stejným způsobem. Jednotlivé sekce pro určité pohlaví jsou od sebe odděleny tečkovanou čárou. Čára je vykreslena za určitou sekcí pouze tehdy, kdy je nalezen objekt následující sekce (v případě, že by nebyly nalezeny objekty určitého pohlaví, nebyly by vykresleny redundantní čáry). Vertikální čára je vykreslena v místě pod právě vykreslenou sekcí.

Modelování rodokmenu Další z možných modelů dostupných k vygenerování je rodokmen, kterým zpravidla rodinné pátrání začíná [5].



Obrázek 6.14: Screenshot modelu rodokmenu

Struktury rodokmenu Pro účely generování rodokmenu bylo nutné vytvořit další dodatečné třídy, které slouží výlučně k modelování rodokmenu. Tyto třídy jsou *RodokmenClovek* a *RodokmenData*.

RodokmenClovek třída reprezentuje jednu osobu, která bude součástí generovaného modelu. Obsahuje v první řadě odkaz na danou osobu typu *JmenoPrijmeni*, dále obsahuje atributy otec, matka, partner typu *RodokmenClovek* a řetězec deti typu *RodokmenClovek.* Tato struktura dále umožní vykreslit rodokmen stylem preorder.

Všechny objekty typu *RodokmenClovek* jsou ukládány do třídy vzoru singleton *RodokmenData.* Tato třída obsahuje instanci na sebe sama. Dále se v ní nachází seznam objektů typu *RodokmenClovek* rodokmenLidi, který obsahuje všechny objekty dále používáne při modelování rodokmenu.

Iniciování potřebných objektů Po definování potřebných tříd a struktur je nutné tyto objekty vytvořit a naplnit potřebnými údaji. Tento úkol je proveden postupným procházením všech objektů v seznamu GuiObjectsFiltered, vytvořením pro každý tento objekt instanci třídy *RodokmenClovek* a naplněním jejich atributů. Pro každý *GUIObject* je získán příslušný *RodokmenClovek* objekt, o toto se stará *RodokmenData* funkce *getClovekById().* Tato funkce, v případě existence příslušné *RodokmenClovek* instance, vrátí tuto instanci, nebo ji sama vytvoří a rovnou přidá do svého seznamu existujících lidí. Po získání této instance znovu iterujeme nad všemi objekty v GuiObjectsFiltered a podle rolí, které má člověk přiřazené (z plochy přípravy objektů), hledáme a přiřazujeme jeho rodinné příslušníky. Všechny tyto role předpokládají pouze jednoho partnera, otce a matku, s libovolným počtem potomků.

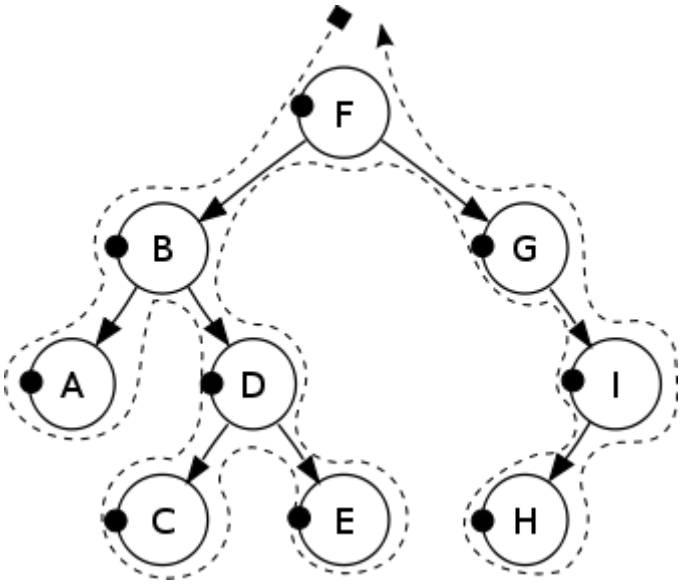
Pokud má člověk přiřazenou roli:

- **ženicha**
  - jediného rodinného příslušníka, kterého můžeme přiřadit, je partner (v případě, že nevěsta má přiřazenou roli nevěsty) – hledáme svatby, ve kterých člověk figuruje jako ženich a iterátor jako nevěsta
  - bereme v potaz, že nevěsta mohla někdy v průběhu života změnit své jméno, takže v případě, že nenajdeme nevěstu, se díváme do tabulky změna jmen pro ID iterátoru
- **nevěsty**
  - podobně jako ženich, jediný rodinný příslušník, kterého lze přiřadit, je partner (v případě, že ženich má přiřazenou roli ženicha) – hledáme svatby, ve kterých člověk figuruje jako nevěsta a iterátor jako ženich
  - dále nevěsta znovu mohla změnit své jméno, takže v případě, že nenajdeme ženicha, se díváme do tabulky změna jmen pro ID právě zpracovávané osoby
- **otce** – můžeme přiřadit:
  - partnera – hledáme záznam narození, ve kterém figuruje daný člověk jako otec narozeného a iterátor jako matka narozeného
  - děti – v případě, že mají přiřazenou roli potomka
    - hledáme záznam narození, ve kterém figuruje daný člověk jako otec a iterátor jako narozený
    - hledáme záznam svatby, ve kterém figuruje daný člověk jako otec ženicha a iterátor jako ženich
    - hledáme záznam svatby, ve kterém figuruje daný člověk jako otec nevěsty a iterátor jako nevěsta
- **matky** – můžeme přiřadit:
  - partnera – hledáme záznam narození, ve kterém figuruje daný člověk jako matka narozeného a iterátor jako otec narozeného
  - děti – v případě, že mají přiřazenou roli potomka, hledáme záznam narození, ve kterém figuruje daný člověk jako matka a iterátor jako narozený
  - ve všech těchto případech počítáme s tím, že matka mohla kdykoliv změnit své jméno, a díváme do tabulky změna jmen pro ID právě zpracovávané osoby
- **potomka** – můžeme přiřadit:
  - otce
    - hledáme záznam narození, ve kterém figuruje daný člověk jako narozený a iterátor jako otec
    - hledáme záznam svatby, ve které figuruje daný člověk jako ženich a iterátor jako otec ženicha

    - hledáme záznam svatby, ve které figuruje daný člověk jako nevěsta a iterátor jako otec nevěsty
- matku
    - hledáme záznam narození, ve kterém figuruje daný člověk jako narozený a iterátor jako matka narozeného
    - matka také mohla někdy v průběhu života změnit své jméno, takže v případě, že nenajdeme nevěstu, se díváme do tabulky změna jmen pro ID iterátoru

Generování modelu Po definování a iniciování všech objektů potřebných pro vygenerování rodokmenu, zbývá již jen vygenerovat samotný model.

Model bude vykreslován pomocí rekurzivního algoritmu procházení binárním stromem stylem preorder. Tento algoritmus je vcelku přímočarý – navštívíme uzel, zpracujeme tento uzel a postupně procházíme jeho potomky stejným způsobem.



Obrázek 6.15: Peorder algoritmus procházení stromu, obrázek převzat z [10]

Vykreslení grafických prvků Před popsáním algoritmu budou stručně rozebrány postupy při vykreslování grafických prvků.

Obdélník

Vykreslení objektu reprezentujícího člověka probíhá podobně jako je v této aplikaci zvykem. Je vykreslen obdélník, před který je vypsáno jméno daného člověka. Tomuto obdélníku je přiřazena barva stejným způsobem jako v přípravě objektů, nebo při generování objektů podle pohlaví (Pokud člověk někdy figuroval jako otec nebo ženich – modrá barva, matka nebo nevěsta - červená barva, jinak světle zelená).

Propojovací čára
Jednotlivé objekty, které mají příbuzenské vztahy, jsou mezi sebou propojeny tečkovanou čárou. Souřadnice jsou nastaveny podle obdélníků, které určitá čára propojuje.

- Start vykreslování čáry
  - Souřadnice x – souřadnice x prvního obdélníku + půlka šířky obdélníku
  - Souřadnice y – souřadnice y prvního obdélníku + půlka výšky obdélníku
- Konec vykreslování čáry
  - Souřadnice x – souřadnice x druhého obdélníku + půlka šířky obdélníku
  - Souřadnice y – souřadnice y druhého obdélníku + půlka výšky obdélníku

První kroky První krok algoritmu vygenerování rodokmenu je určení kořenů stromu, aneb prvky, které nemají žádného předchůdce. V naší struktuře je to ten člověk, který nemá známého otce, ani matku. Před generováním modelu se uloží do proměnné hodnota x a y, která se předá rekurzivní preorder funkci, aby bylo vždy jasné, kde v prostoru se nacházíme. Dále je jako parametr předán seznam objektů typu Line reprezentující tečkovanou propojovací čáru mezi objekty, do kterého se budou postupně ukládat všechny čáry na vykreslení. Po deklarování proměnných reprezentujících seznam spojujících čar a souřadnic se v cyklu prochází všechny objekty v GUIObjectsFiltered, vytvoří se seznam objektů typu *RodokmenClovek,* který se uloží do v *RodokmenData* , a tento seznam lidí se v cyklu začne procházet.

Rekurzivní preorder algoritmus Jak narazíme na objekt, který nemá otce, ani matku, víme, že jsme našli kořenový objekt. Nad tímto objektem zavoláme funkci *preorderRodokmenData()*, která tento objekt zpracuje a rekurzivně vygeneruje zbytek tohoto stromu.

Na začátku zpracování člověka je test, zda je již tento objekt vykreslen (procházíme všechny objekty a některé z nich, například děti s dvěma rodiči, budou zpracovávány vícekrát), pokud tato situace nastane, zpracování je ukončeno.

Dále je kontrolováno, jestli na souřadnicích, které jsou předány preorder funkci na vykreslení daného objektu člověka, již není vykreslený jiný objekt (sourozenci mají každý více dětí, ...), pokud tato situace nastane, posune se objekt o řadu dolů, následně rekurzivně kontroluje i tyto souřadnice a posouvá se o řádek dolů, dokud nenarazí na volné místo.

Následuje samotné vykreslení grafiky reprezentující objekt člověka. Po vykreslení člověka zjistíme, zdali má přiřazeného partnera, a pokud ano, vykreslíme ho hned vedle zpracovávané osoby a propojíme tyto dva objekty čárou. Následně inicializujeme seznam deti typu *RodokmenClovek*, do kterého si uložíme všechny děti partnera (tyto děti budou vykresleny s propojením k tomu objektu, který bude zpracováván dříve (aneb právě vykreslovaný objekt)). K získaným dětem partnera následně přidáme všechny děti právě zpracovávaného objektu.

Po vykreslení zpracovávaného objektu, vykreslení jeho partnera a získání dětí, zbývá již jen rekurzivně udělat tu samou akci pro každého potomka. Prvním krokem při vykreslování dětí je posunutí souřadnice x doleva (rodiče by měli být co nejvíce uprostřed nad dětmi)

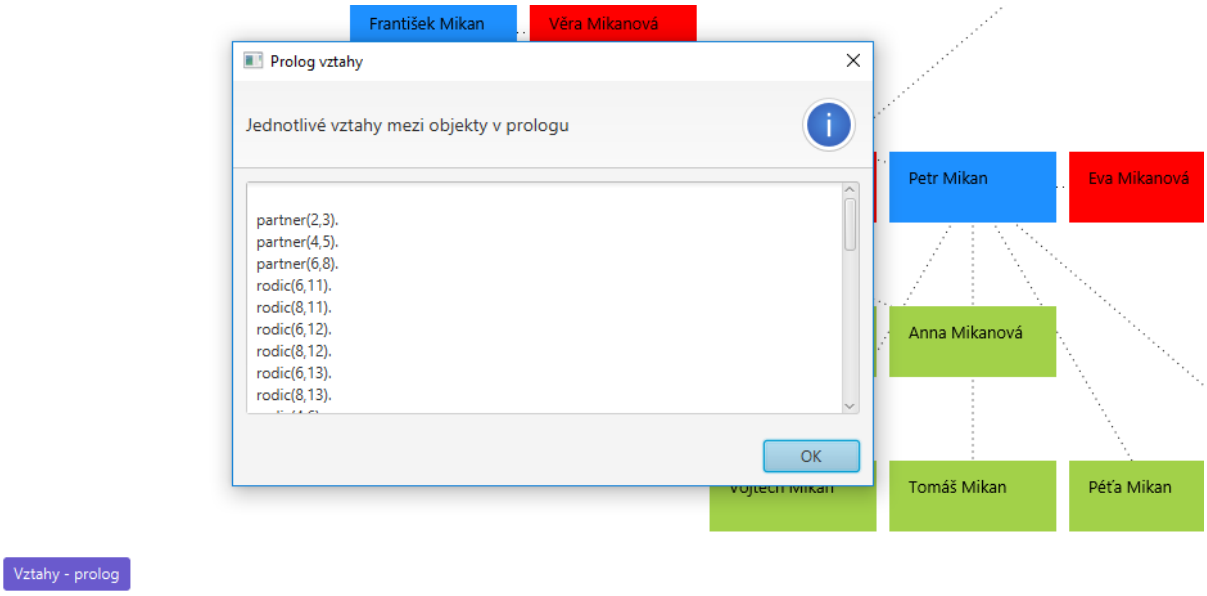
vzhledem k počtu potomků – pokud souřadnice dosáhne levé hranice modelu, nastaví se na 0. Dále pokud má potomek partnera, rezervujeme pro něho místo (posuneme souřadnici x pro dalšího potomka o dvojnásobnou plochu). Potom již zbývá rekurzivně volat funkci *preorderRodokmenData()* na každého potomka a propojit vykreslovaný objekt a každého jeho potomka.

Po projití celého modelu a nastavení grafických prvků (souřadnice a propojovací čáry) se všechny prvky najednou přidají na plochu.

Výstup v prologu Kromě grafického vyobrazení rodokmenu je dále dostupná možnost nechat si vypsat jednotlivé vztahy mezi objekty pomocí zápisu v prologu. Tlačítko na vygenerování je umístěno v levém rohu pod rodokmenem.

V tomto výpisu figurují pouze dva typy vztahů – partner a rodič. Každý partnerský vztah je zapsán jednou, každý rodičovský vztah je zapsán pro každého rodiče (v případě, že víme o matce i otci – budou dva zápisy pro každé dítě s odlišným rodičem).

Tento výpis umožní další případnou analýzu vygenerovaného rodokmenu.



Obrázek 6.16: Screenshot výpisu rodokmen vztahů v prologu

Kapitola 7

Testování

Tato kapitola se bude zabývat problematikou testování této aplikace. Žádná aplikace by nebyla kompletní bez hledání chyb v programu a subjektivních názorů uživatelů, kteří se testování účastnili. Hlavním prvkem, na které se testování soustředilo, bylo grafické uživatelské rozhraní a použitelnost aplikace. V průběhu vytváření se aplikace postupně měnila podle této zpětné vazby jak od testovacích uživatelů, tak podle názorů, které vznikly v čase konzultací s vedoucím práce.

7.1 Průběžné testování aplikace

V průběhu vytváření aplikace, obzvlášť vzhledem k tomu, že byla vyvíjena pouze jedním programátorem, byla testována mnou při každé nové změně. Aplikace byla vyvíjena takovým stylem, že po jakékoliv modifikaci programu byla aplikace proklikána a ověřeno, že nedošlo k vytvoření nějaké neočekávané funkčnosti. Dále byly testovány situace s neočekávanými a mezními hodnotami a ošetřeno, že i za těchto situací se program chová podle očekávání.

7.2 Akceptační testování použitelnosti a grafického rozhraní

Samotné akceptační testování aplikace se provádí až když se aplikace blíží finálnímu stavu. Hlavním cílem tohoto testování je zjištění, zdali program dosáhl požadované kvality a jeho funkcionalita odpovídá potřebám uživatelů [3]. Testování pro tuto aplikaci proběhlo na 5 potencionálních uživatelích aplikace různých věkových skupin. Toto testování se soustředilo převážně na první dojem z aplikace, dobu, po jakou uživateli trvalo provést určitou akci, a jejich subjektivní pocity z této zkušenosti.

Dojmy z tohoto testování byly převážně pozitivní, avšak vedly k určitým barevným úpravám GUI a změnu rozmístění určitých prvků grafického rozhraní. Dále bylo v reakci na výsledky těchto testů pozměněno přiřazování jednotlivých rolí uživatelům z ručního přiřazování rolí (viz. Zadání „navrhněte, jak lze uživatelsky přívětivě tyto role přisuzovat objektům vytvářeného modelu“) na takový stav, kdy má osoba všechny jeho možné role přiřazené a uživatel spíše „odebírá“ role, o které nemá zájem.

7.3 Výsledky testování

Testování určitě přineslo několik cenných poznatků a úprav. Poukázalo se na několik návrhových nedostatků jak grafického rozhraní, tak implementace aplikace. Celkové hodnocení aplikace uživateli bylo pozitivní a tvrdili, že by aplikaci zkusili, kdyby existovala data v elektronické podobě, která by pro ně byla relevantní.

Kapitola 8

Možná rozšíření

V této kapitole budou popsána možná rozšíření a případné nedostatky aplikace, které byly v průběhu vývoje a testování aplikace odhaleny.

8.1 Duplikace jmen

Problém se současnými daty a návrhem zpracování je ten, že jsou automaticky hledána logická propojení mezi osobami pomocí stejného jména a příjmení. Tento přístup je funkční do takové míry, dokud je malé množství duplikací jmen. Byl by možný určitý počet rozšíření, které by mohly tento problém do určité míry snížit, ale i tak by nebyl eliminován (například přidávat aliasy u stejných jmen po určité době od sebe (např. 100 let)). Pokud nebudou data parsována ručně podle mého názoru problém přetrvá.

S tímto problémem jde ruku v ruce nedokonalost dat. Data přepsaná z matrik do excel souboru a dále mnou zpracovaná pomocí python scriptu prodělala z hlediska validity dat nemalou ránu. Jména byla pro účely této aplikace upravována pomocí velké skupiny pravidel, některé z nich specifické pro konkrétní záznamy. To samé bylo nutné provést nad například povoláními, atd.

8.2 Sociální modely

Dalším rozšířením aplikace je samozřejmě jakýkoliv sociální model, který dává smysl vzhledem k dostupným datům. Nabízí se například místa narození, povolání, ... – tyto modely momentálně nejsou vytvořeny vzhledem k malému počtu záznamů, které tyto data obsahují.

8.3 Grafické rozhraní

Věřím, že aktuální grafické rozhraní je dostačující pro účely této aplikace. I tak byla tato aplikace mou první aplikací v JavaFX a jsem si jistý, že někdo s většími zkušenostmi by dokázal celkový grafický návrh zjednodušit a zefektivnit.

Kapitola 9

Závěr

Cílem této bakalářské práce nebylo jen vytvoření aplikace, která nějakým způsobem zpracuje poskytnutá data z matrik a archiválií a vytváří nad těmito daty různé sociální modely, ale také vyzkoušení a experiment, zda je něco takového vůbec nad těmito daty možnou a smysluplnou činností. V první řadě bylo nutné seznámit se s programovacím jazykem Java a stylem, jakým se vytváří aplikace v JavaFX. Snažil jsem se nastudovat co největší množství systémů na zpracování alespoň podobných dat z archiválií, ale bohužel takové systémy neexistují ve velkém množství. Dále bylo nutné vymyslet, jakým způsobem zpracovat poskytnutá data, vzhledem k jejich nespolehlivé povaze. I přes tyto překážky bych si dovolil konstatovat, že aplikace funguje podle představ a dokáže nasimulovat určité modely, má-li dostatečně přesná a korektní data, jak jsem dokázal nad vlastním rodokmenem. Na závěr jsem svojí implementovanou aplikaci otestoval a podle zpětné vazby upravil a navrhl možná rozšíření do budoucna.

Výsledkem této práce je desktopová GUI aplikace čerpající data z MySQL databáze, která umožňuje přívětivě přisuzovat role jednotlivým osobám a modelovat určité sociální modely nad vybranými daty. Aplikace je, alespoň podle uživatelů, na kterých byla aplikace testována, uživatelsky přívětivá a jednoduchá na ovládání, na což byl při vývoji kladen důraz.

Literatura

[1] Doležal, M.: *Ancestry (Rodokmen)*. [Online; navštíveno 20.04.2018]. Retrieved from: https://www.slunecnice.cz/sw/rodokmen/

[2] Herout, P.: *Java - Učebnice jazyka*. Kopp. 2010. ISBN 978-80-7232-398-2.

[3] Hlava, T.: *Fáze a úrovně provádění testů*. 2011. [Online; navštíveno 20.04.2018]. Retrieved from: http://testovanisoftwaru.cz/tag/akceptacni-testovani/

[4] Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. Elsevier Science & Technology. 2010. ISBN 978-0-12-375030-3.

[5] Lednická, B.: *Sestavte si rodokmen*. Grada. 2012. ISBN 978-80-247-4069-0.

[6] MyHeritage Ltd.: *MyHeritage*. [Online; navštíveno 20.04.2018]. Retrieved from: https://www.myheritage.cz/

[7] Oracle: *What Is JavaFX?* [Online; navštíveno 20.04.2018]. Retrieved from: https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

[8] Čápka, D.: *Popis MVC architektury*. 2012. [Online; navštíveno 20.04.2018]. Retrieved from: https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury

[9] The Griffon Team: *MVC Patterns*. [Online; navštíveno 20.04.2018]. Retrieved from: http://griffon-framework.org/tutorials/5_mvc_patterns.html

[10] Wikipedia contributors: *Tree traversal*. [Online; navštíveno 20.04.2018]. Retrieved from: en.wikipedia.org/wiki/Tree_traversal

[11] Willems, K.: *Python Excel Tutorial: The Definitive Guide*. [Online; navštíveno 20.04.2018]. Retrieved from: https://www.datacamp.com/community/tutorials/python-excel-tutorial

[12] www.mysqltutorial.org: *What Is MySQL and Why It Is the World's Most Popular Open Source Database*. [Online; navštíveno 20.04.2018]. Retrieved from: http://www.mysqltutorial.org/what-is-mysql/

Příloha A

Jak zprovoznit tuto aplikaci

V této příloze budou popsány akce, které jsou nutné k úspěšném spuštění aplikace.

Konfigurace dat

V odevzdaných souborech bude k nalezení adresář "data". V tomto adresáři jsou nahrány soubory:

xmikandata.xlsx Datový soubor v excelu - z tohoto souboru byla brána veškerá data používaná v programu.

parse.py Parsovací script, který se připojí na databázi a naplní jí požadovanými rozparsovanými daty ze souboru xmikandata.xlsx.

xmikan00.sql Script v jazyce SQL, který vytvoří tabulky a naplní je požadovanými daty.

Databáze Pro korektní fungování aplikace a scriptů je nutné mít nejprve spuštěný nějaký databázový systém MySQL. Aplikace i parse.py script se připojí na:

- localhost:3306
- db: "xmikan00"
- user: "root"
- heslo: ""

Na zprovoznění aplikace stačí pouze SQL script xmikan00.sql, který obsahuje jak strukturu tabulek, tak samotná data. Vše, co je potřeba, je vytvořit databázi se jménem "xmikan00"a importovat tento SQL script.

Spuštění aplikace

Po vytvoření databáze a naplnění daty je možné spustit danou aplikaci. Samotná aplikace je přeložena do souboru "BP_xmikan00.jar", který je v rootu adresáře. Tento soubor stačí spustit dvojklikem (je samozřejmě nutné mít nainstalované ideálně nejnovější Java Runtime Environment).